INDEX

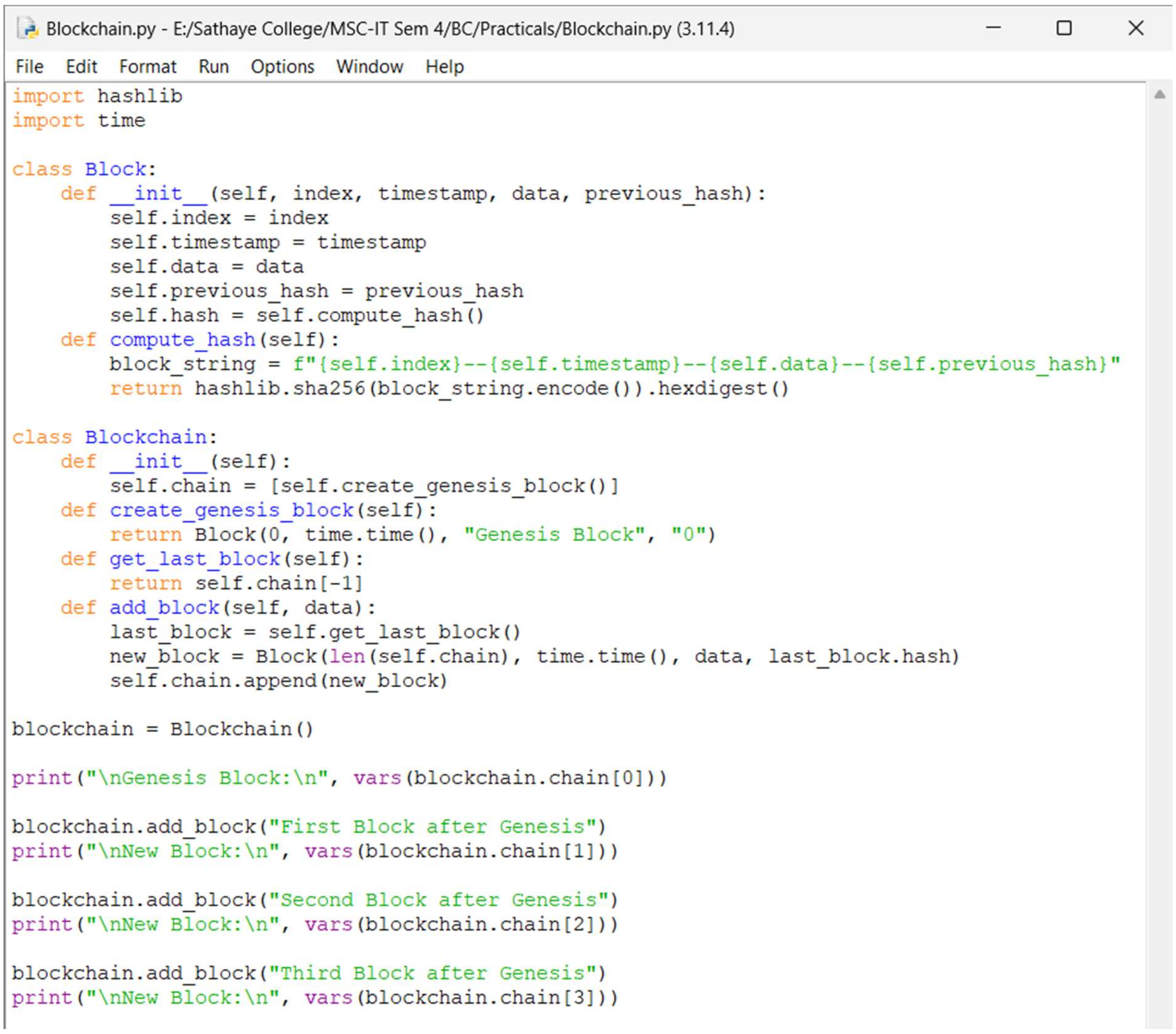
Sr. No	Date	Practical Title	Sign
1		Create a blockchain and a genesis block and execute it.	
2		Implement and demonstrate the use of solidity programming. a. Counter b. Calculator c. Increment/Decrement Operator	
3		Loops in solidity a. For loop b. While loop	
4		Arrays in solidity.	
5		Operators in solidity a. Comparison operator (==, !=) b. Logical operator (&&, ||) c. Assignment operator (+=, -=, *=, /=) d. Ternary operator (?:)	
6		Mathematical functions (mulmod and addmod) and Function overloading.	
7		Implementation of interface and inheritance.	
8		Selection of candidate in election.	
9		Write a solidity program to create an array of roll no’s and then create a smart contract where it checks the values of the roll no and perform AND operation with today’s date DD and if the result is even, then display the message “Student is allowed” else “Denied”. (Optional)	
10		Write a solidity program to find the sum of an array of 10 numbers which are taken from the user and then create a smart contract to find the AND operation of Odd positioned numbers and OR operation of Even positioned numbers including 0th Index, hence find the product of the result and also identify whether the result is part of the array or not. (Optional)	
11		Write a solidity program to find whether a number is even or odd and another number is prime or composite. Also find the AND and OR operation of the two numbers. (Optional)	

SACHIN BHOJA ACHARYA
Roll No: 01

Practical No 1

AIM : Create a blockchain and a genesis block and execute it.

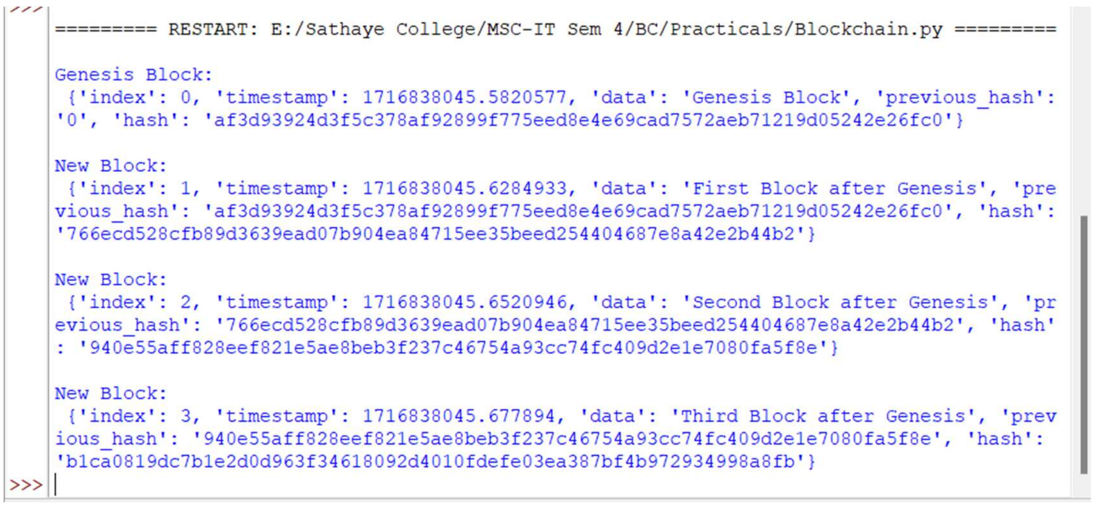
CODE :



Blockchain.py - E:/Sathaye College/MSC-IT Sem 4/BC/Practicals/Blockchain.py (3.11.4)
File Edit Format Run Options Window Help

```python
import hashlib
import time

class Block:
    def __init__(self, index, timestamp, data, previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.compute_hash()
    def compute_hash(self):
        block_string = f"{self.index}--{self.timestamp}--{self.data}--{self.previous_hash}"
        return hashlib.sha256(block_string.encode()).hexdigest()

class Blockchain:
    def __init__(self):
        self.chain = [self.create_genesis_block()]
    def create_genesis_block(self):
        return Block(0, time.time(), "Genesis Block", "0")
    def get_last_block(self):
        return self.chain[-1]
    def add_block(self, data):
        last_block = self.get_last_block()
        new_block = Block(len(self.chain), time.time(), data, last_block.hash)
        self.chain.append(new_block)

blockchain = Blockchain()

print("\nGenesis Block:\n", vars(blockchain.chain[0]))

blockchain.add_block("First Block after Genesis")
print("\nNew Block:\n", vars(blockchain.chain[1]))

blockchain.add_block("Second Block after Genesis")
print("\nNew Block:\n", vars(blockchain.chain[2]))

blockchain.add_block("Third Block after Genesis")
print("\nNew Block:\n", vars(blockchain.chain[3]))
```

OUTPUT :



```
========= RESTART: E:/Sathaye College/MSC-IT Sem 4/BC/Practicals/Blockchain.py =========

Genesis Block:
 {'index': 0, 'timestamp': 1716838045.5820577, 'data': 'Genesis Block', 'previous_hash':
'0', 'hash': 'af3d93924d3f5c378af92899f775eed8e4e69cad7572aeb71219d05242e26fc0'}

New Block:
 {'index': 1, 'timestamp': 1716838045.6284933, 'data': 'First Block after Genesis', 'pre
vious_hash': 'af3d93924d3f5c378af92899f775eed8e4e69cad7572aeb71219d05242e26fc0', 'hash':
'766ecd528cfb89d3639ead07b904ea84715ee35beed254404687e8a42e2b44b2'}

New Block:
 {'index': 2, 'timestamp': 1716838045.6520946, 'data': 'Second Block after Genesis', 'pr
evious_hash': '766ecd528cfb89d3639ead07b904ea84715ee35beed254404687e8a42e2b44b2', 'hash'
: '940e55aff828eef821e5ae8beb3f237c46754a93cc74fc409d2e1e7080fa5f8e'}

New Block:
 {'index': 3, 'timestamp': 1716838045.677894, 'data': 'Third Block after Genesis', 'prev
ious_hash': '940e55aff828eef821e5ae8beb3f237c46754a93cc74fc409d2e1e7080fa5f8e', 'hash':
'b1ca0819dc7b1e2d0d963f34618092d4010fdefe03ea387bf4b972934998a8fb'}
>>> |
```

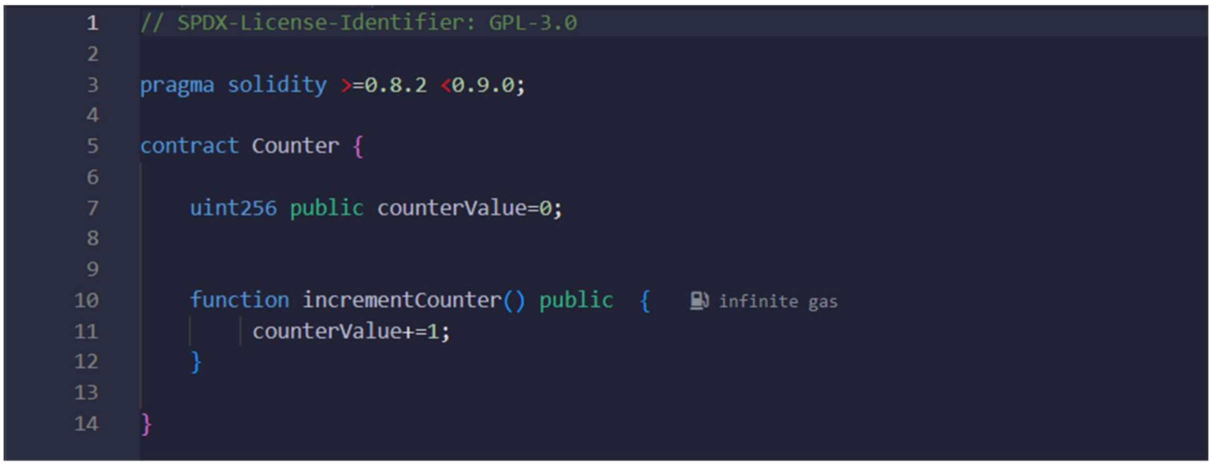
Practical No 2

AIM : Implement and demonstrate the use of solidity programming.

a. Counter
b. Calculator
c. Increment/Decrement Operator

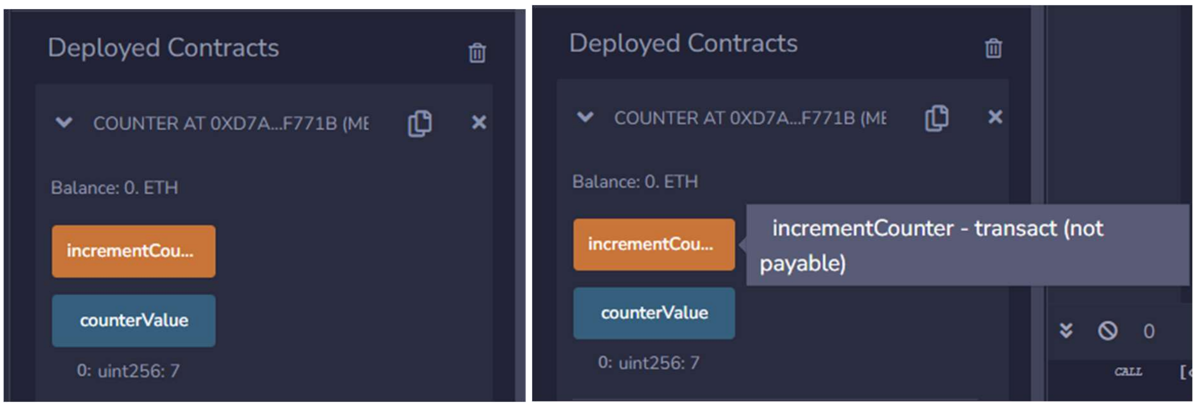
[A] Counter

CODE :



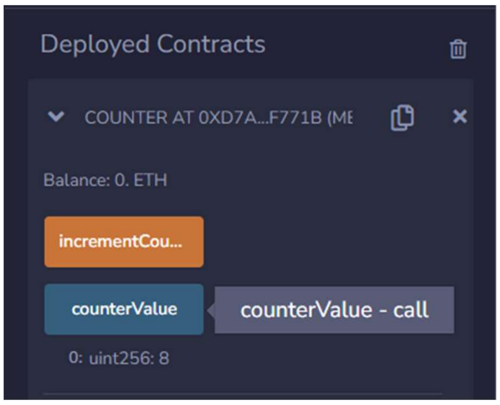
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract Counter {

    uint256 public counterValue=0;


    function incrementCounter() public  {    infinite gas
        counterValue+=1;
    }

}
```

OUTPUT :



Deployed Contracts
COUNTER AT 0XD7A...F771B (ME
Balance: 0. ETH
incrementCou...
counterValue
0: uint256: 7

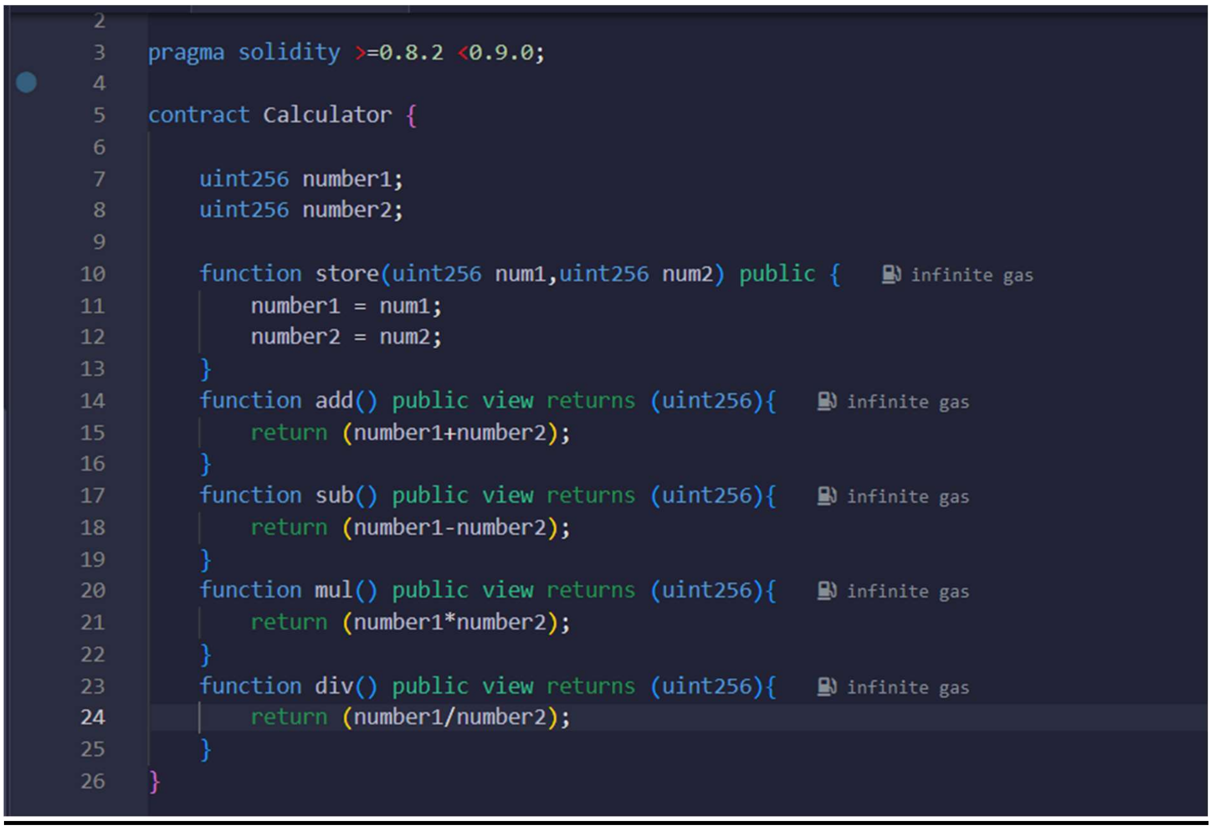
Deployed Contracts
COUNTER AT 0XD7A...F771B (ME
Balance: 0. ETH
incrementCou...
incrementCounter - transact (not payable)
counterValue
0: uint256: 7



Deployed Contracts
COUNTER AT 0XD7A...F771B (ME
Balance: 0. ETH
incrementCou...
counterValue
counterValue - call
0: uint256: 8

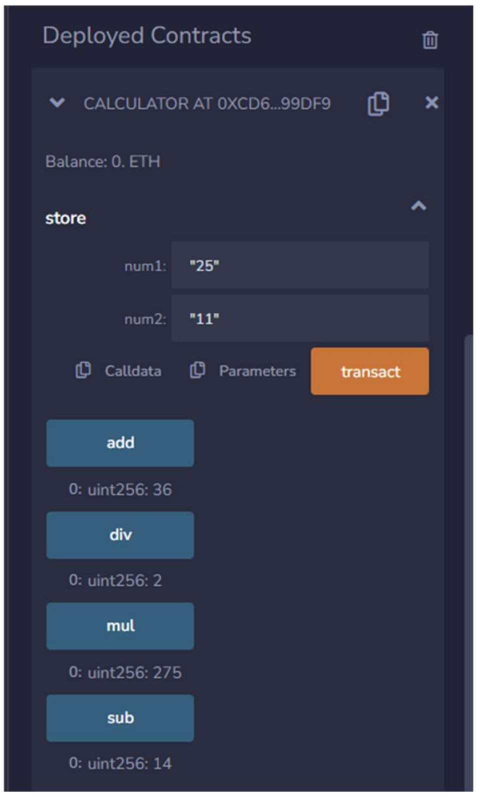
[B] Calculator

CODE :



```
pragma solidity >=0.8.2 <0.9.0;

contract Calculator {

    uint256 number1;
    uint256 number2;

    function store(uint256 num1,uint256 num2) public {    infinite gas
        number1 = num1;
        number2 = num2;
    }
    function add() public view returns (uint256){    infinite gas
        return (number1+number2);
    }
    function sub() public view returns (uint256){    infinite gas
        return (number1-number2);
    }
    function mul() public view returns (uint256){    infinite gas
        return (number1*number2);
    }
    function div() public view returns (uint256){    infinite gas
        return (number1/number2);
    }
}
```

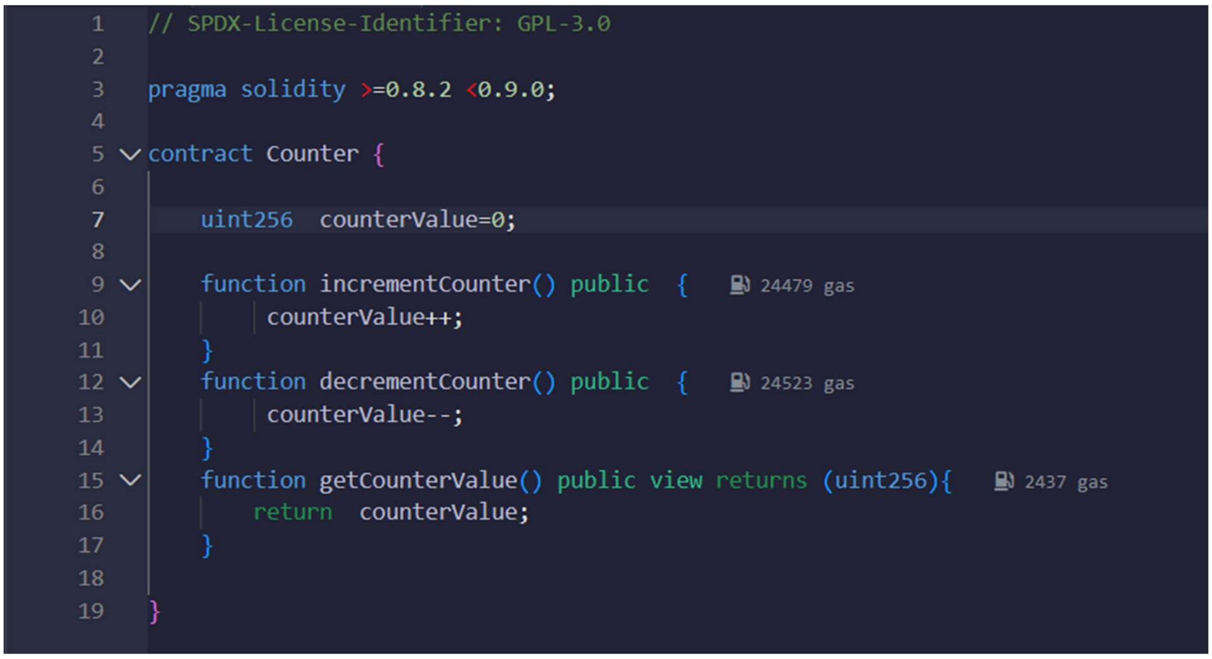
OUTPUT:



```
Deployed Contracts
CALCULATOR AT 0XCD6...99DF9
Balance: 0. ETH
store
num1: "25"
num2: "11"
Calldata  Parameters  transact
add
0: uint256: 36
div
0: uint256: 2
mul
0: uint256: 275
sub
0: uint256: 14
```

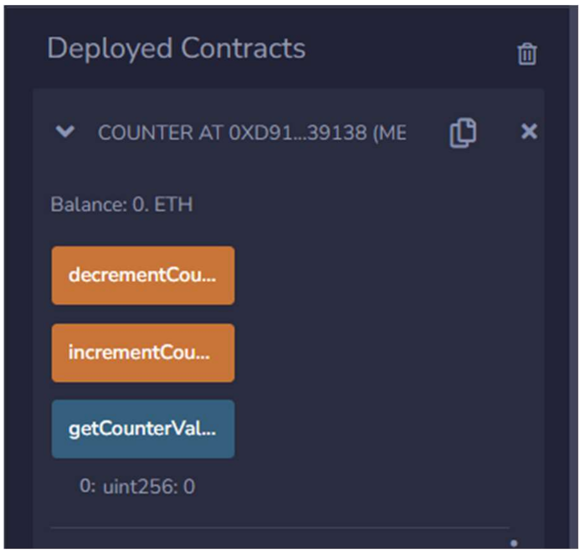
[C] Increment/Decrement Operator

<u>CODE</u> :



```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract Counter {

    uint256  counterValue=0;

    function incrementCounter() public  {     24479 gas
        counterValue++;
    }
    function decrementCounter() public  {     24523 gas
        counterValue--;
    }
    function getCounterValue() public view returns (uint256){     2437 gas
        return  counterValue;
    }

}
```

<u>OUTPUT</u> :

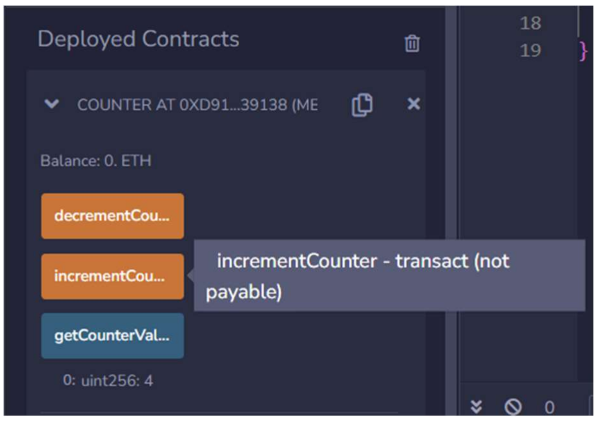


Deployed Contracts

COUNTER AT 0XD91...39138 (ME

Balance: 0. ETH

0: uint256: 0



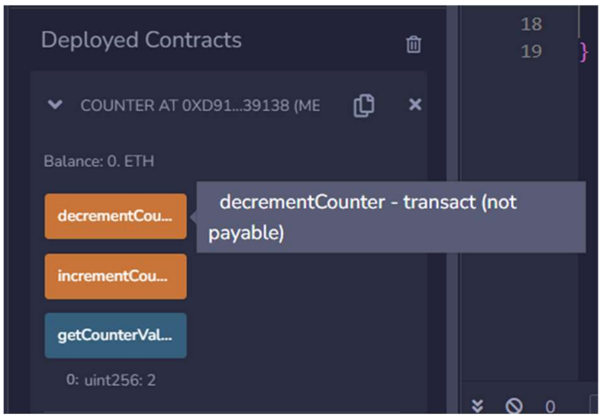
Deployed Contracts

COUNTER AT 0XD91...39138 (ME

Balance: 0. ETH

incrementCounter - transact (not payable)

0: uint256: 4



Deployed Contracts

COUNTER AT 0XD91...39138 (ME

Balance: 0. ETH

decrementCounter - transact (not payable)

0: uint256: 2

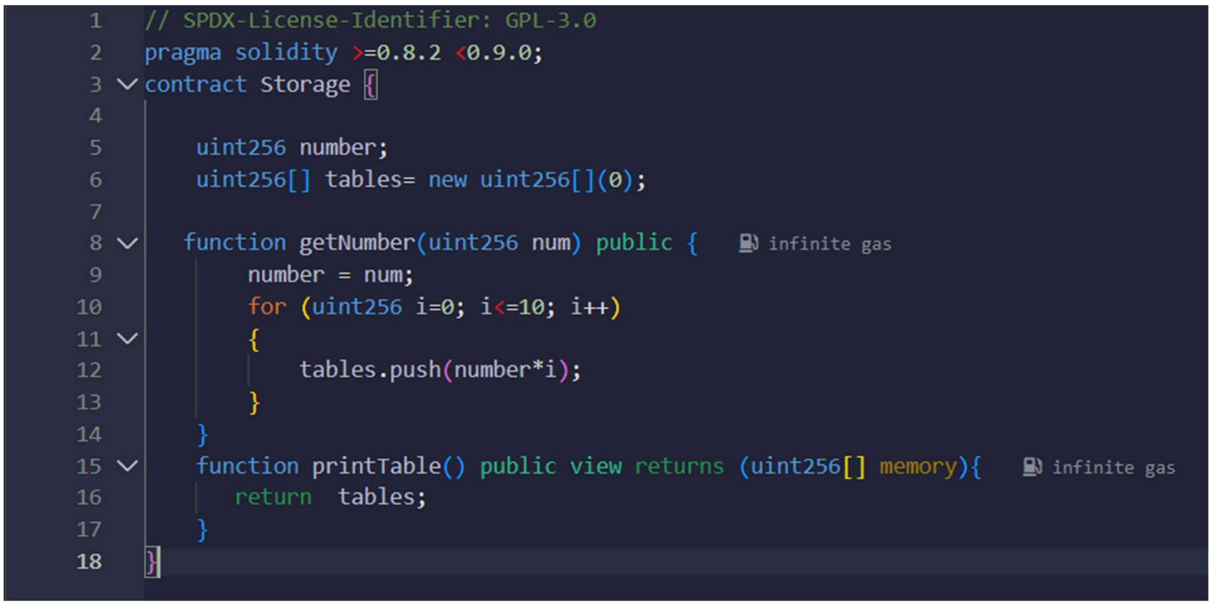
Practical No 3

AIM : Loops in solidity.

a. For Loop
b. While Loop

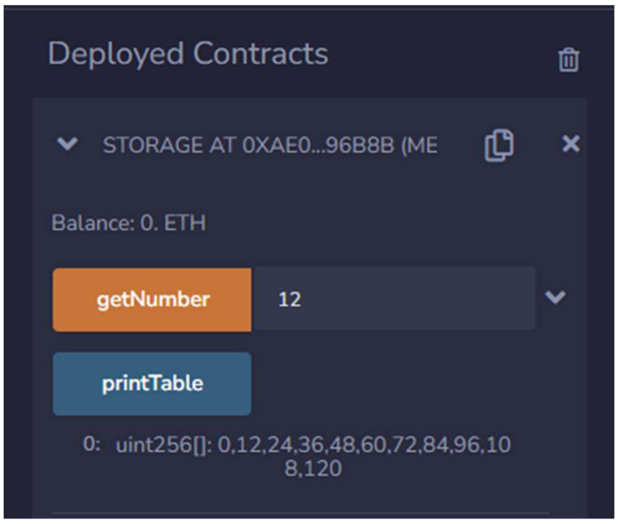
[A] FOR Loop

CODE :



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract Storage {

    uint256 number;
    uint256[] tables= new uint256[](0);

   function getNumber(uint256 num) public {    infinite gas
        number = num;
        for (uint256 i=0; i<=10; i++)
        {
            tables.push(number*i);
        }
    }
    function printTable() public view returns (uint256[] memory){    infinite gas
       return  tables;
    }
}
```

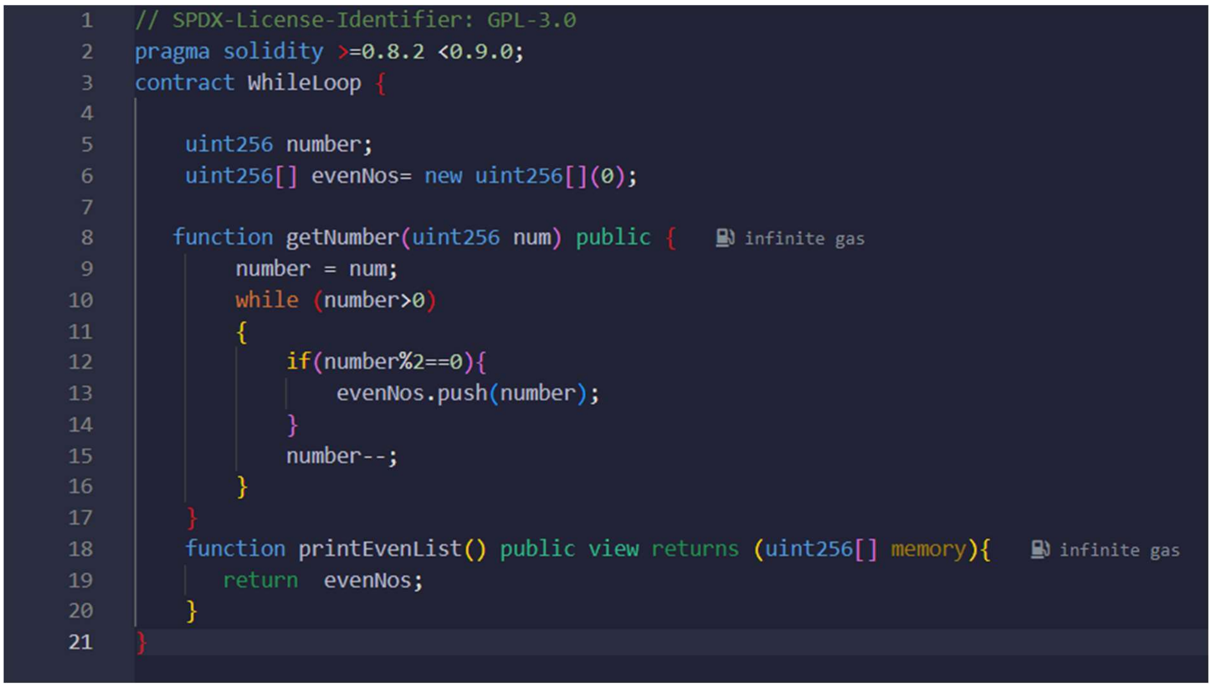
OUTPUT :



```
Deployed Contracts

STORAGE AT 0XAE0...96B8B (ME

Balance: 0. ETH

getNumber  12

printTable

0:  uint256[]: 0,12,24,36,48,60,72,84,96,10
               8,120
```

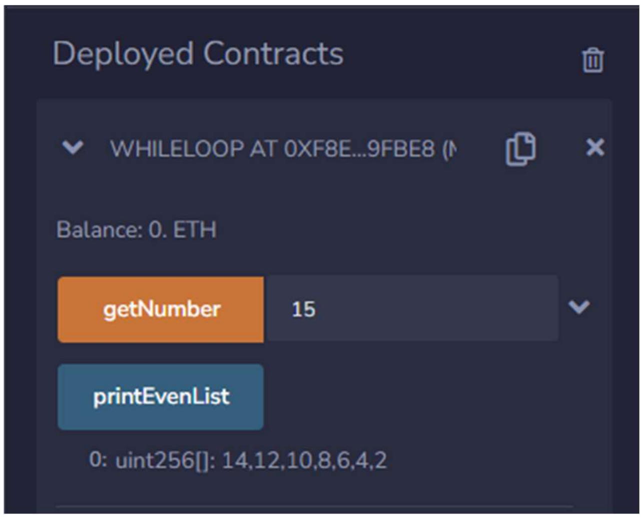
[B] WHILE Loop

<u>CODE</u> :



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract WhileLoop {

    uint256 number;
    uint256[] evenNos= new uint256[](0);

   function getNumber(uint256 num) public {    infinite gas
        number = num;
        while (number>0)
        {
            if(number%2==0){
                evenNos.push(number);
            }
            number--;
        }
    }
    function printEvenList() public view returns (uint256[] memory){    infinite gas
       return  evenNos;
    }
}
```

<u>OUTPUT</u> :

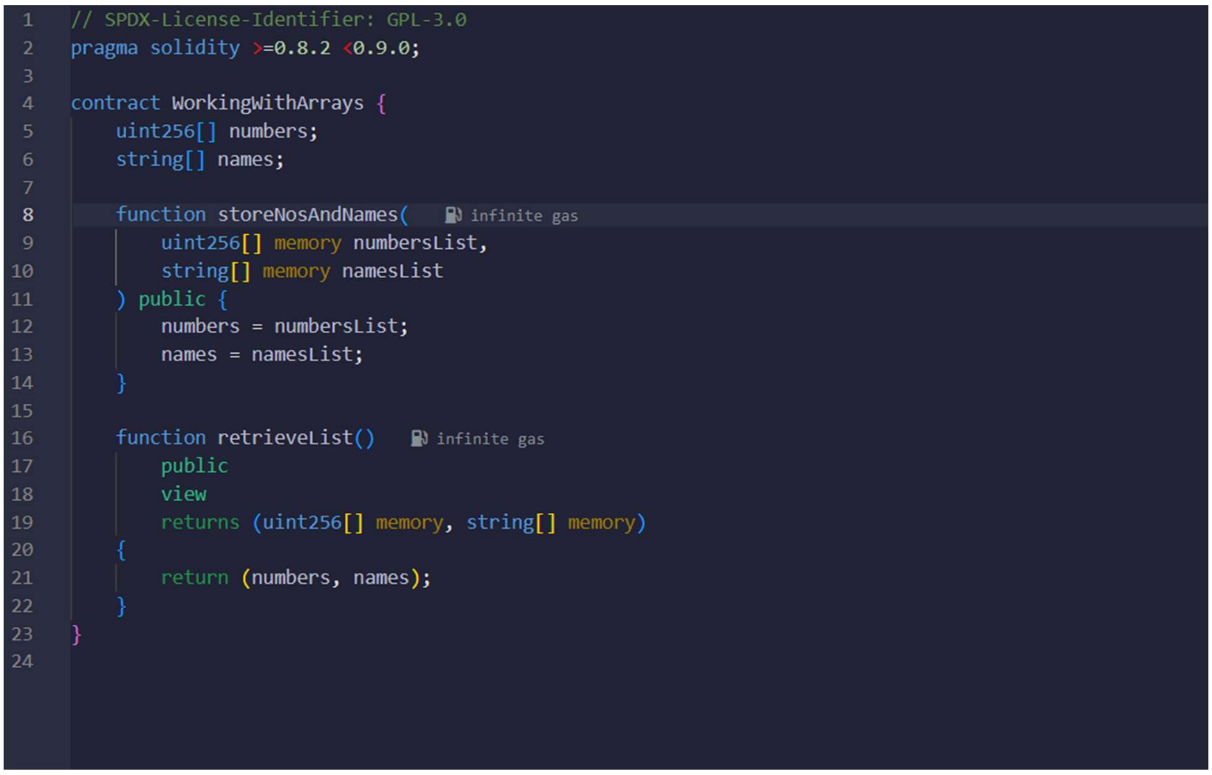


```
Deployed Contracts

WHILELOOP AT 0XF8E...9FBE8 (

Balance: 0. ETH

getNumber  15

printEvenList

0: uint256[]: 14,12,10,8,6,4,2
```

Practical No 4

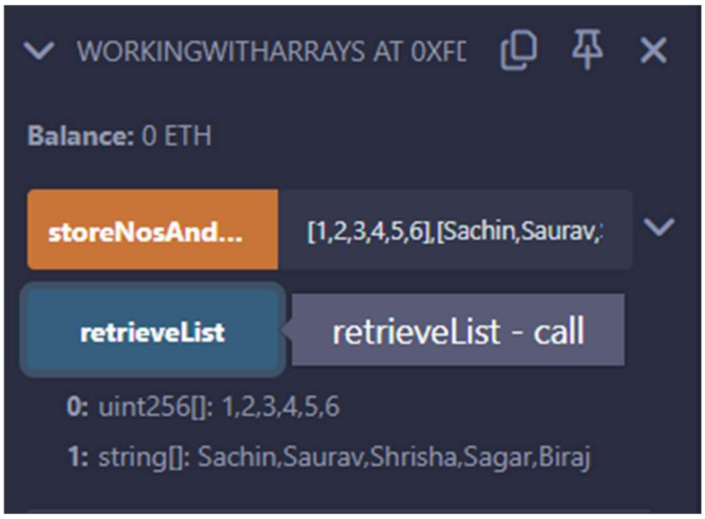
AIM : Arrays in solidity.

CODE :



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;

contract WorkingWithArrays {
    uint256[] numbers;
    string[] names;

    function storeNosAndNames(      infinite gas
        uint256[] memory numbersList,
        string[] memory namesList
    ) public {
        numbers = numbersList;
        names = namesList;
    }

    function retrieveList()     infinite gas
        public
        view
        returns (uint256[] memory, string[] memory)
    {
        return (numbers, names);
    }
}
```

OUTPUT :



```
WORKINGWITHARRAYS AT 0XFE

Balance: 0 ETH

storeNosAnd...  [1,2,3,4,5,6],[Sachin,Saurav,
retrieveList    retrieveList - call

0: uint256[]: 1,2,3,4,5,6
1: string[]: Sachin,Saurav,Shrisha,Sagar,Biraj
```

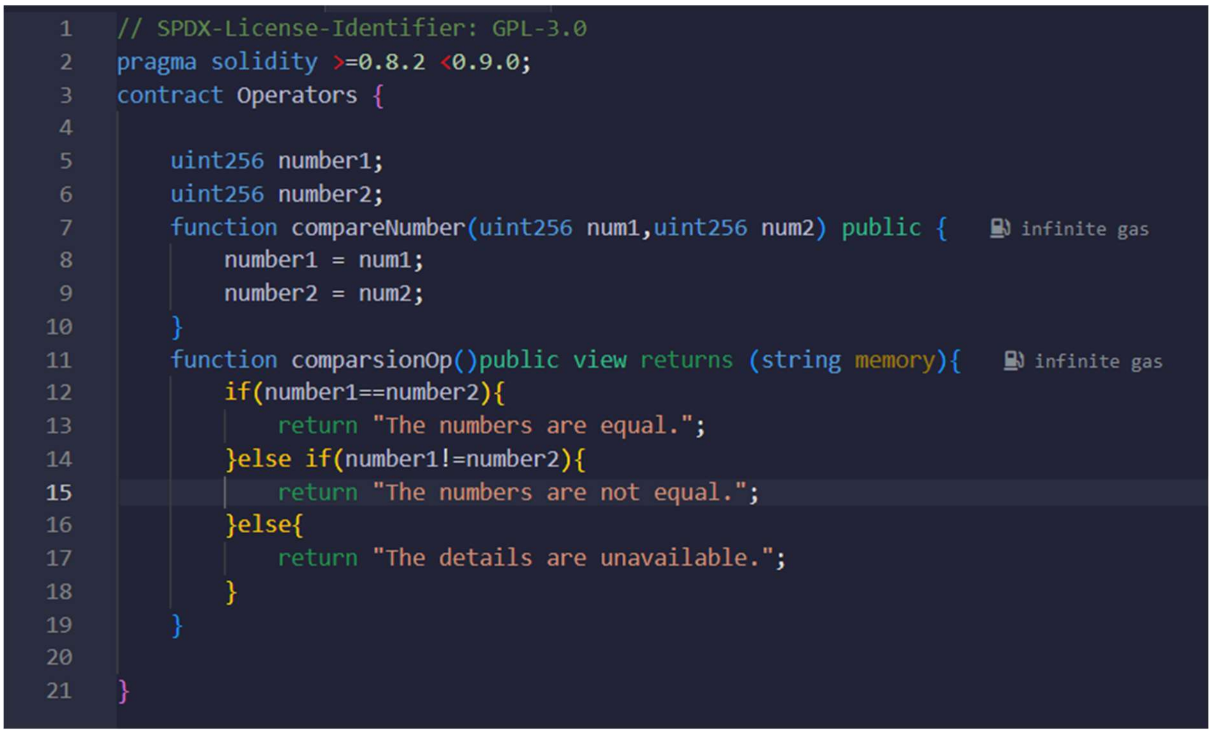
Practical No 5

AIM : Operators in solidity.

a. Comparison Operator (==, !=)
b. Logical operator (&&, ||, !)
c. Assignment operator (+=, -=, *=, /=)
d. Ternary operator (? :)

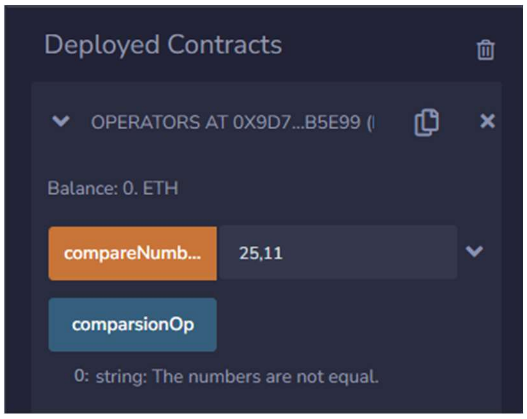
[A] Comparison Operator (==, !=)

CODE :

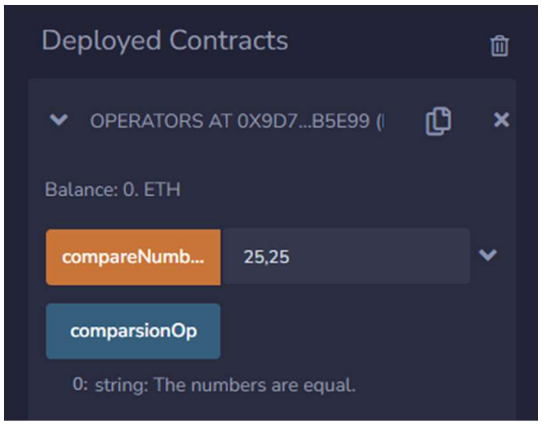


```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract Operators {

    uint256 number1;
    uint256 number2;
    function compareNumber(uint256 num1,uint256 num2) public {      infinite gas
        number1 = num1;
        number2 = num2;
    }
    function comparsionOp()public view returns (string memory){      infinite gas
        if(number1==number2){
            return "The numbers are equal.";
        }else if(number1!=number2){
            return "The numbers are not equal.";
        }else{
            return "The details are unavailable.";
        }
    }

}
```

OUTPUT :



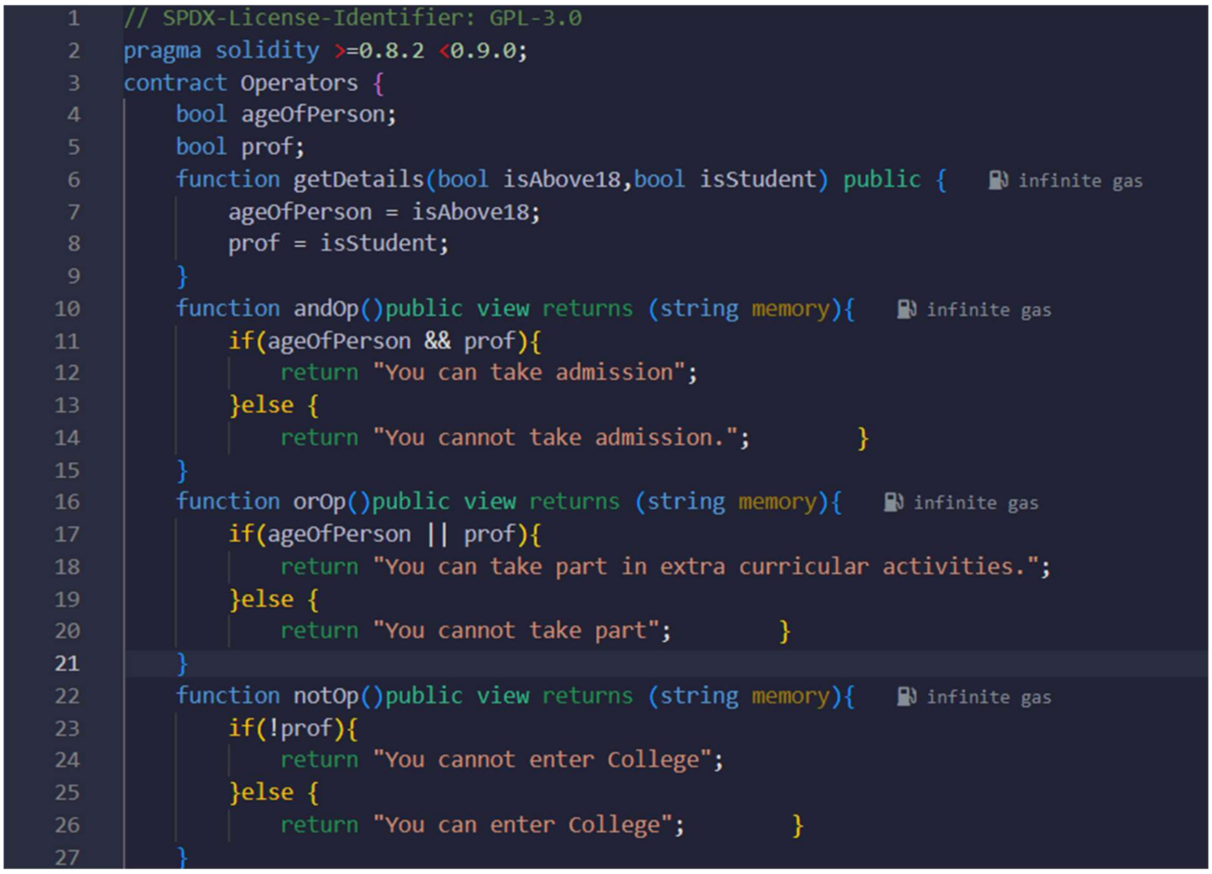
```
Deployed Contracts
OPERATORS AT 0X9D7...B5E99 (
Balance: 0. ETH
compareNumb...  25,11
comparsionOp
0: string: The numbers are not equal.
```



```
Deployed Contracts
OPERATORS AT 0X9D7...B5E99 (
Balance: 0. ETH
compareNumb...  25,25
comparsionOp
0: string: The numbers are equal.
```

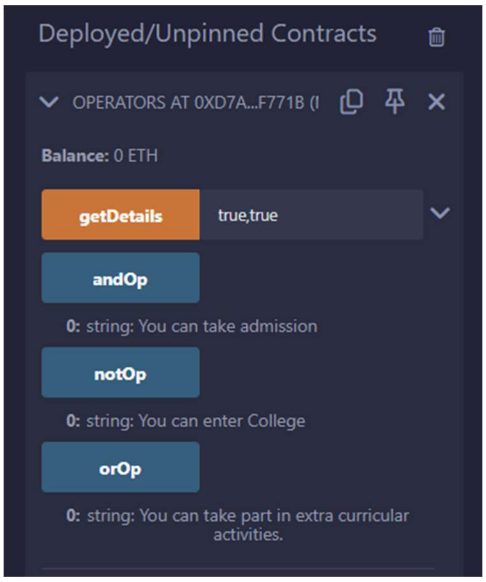
[B] Logical operator (&&, ||, !)

CODE:

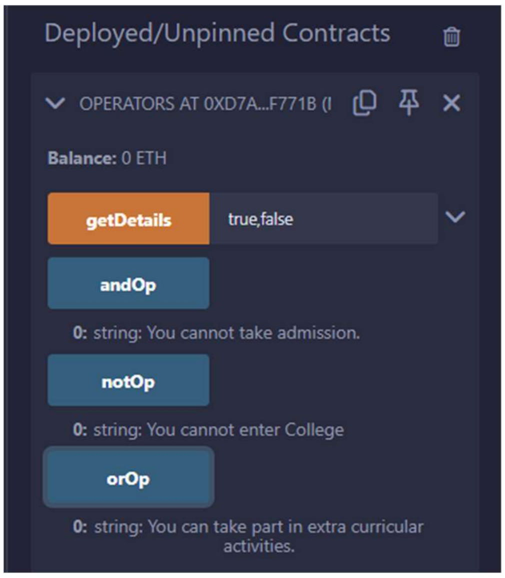


```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract Operators {
    bool ageOfPerson;
    bool prof;
    function getDetails(bool isAbove18,bool isStudent) public {    infinite gas
        ageOfPerson = isAbove18;
        prof = isStudent;
    }
    function andOp()public view returns (string memory){    infinite gas
        if(ageOfPerson && prof){
            return "You can take admission";
        }else {
            return "You cannot take admission.";        }
    }
    function orOp()public view returns (string memory){    infinite gas
        if(ageOfPerson || prof){
            return "You can take part in extra curricular activities.";
        }else {
            return "You cannot take part";        }
    }
    function notOp()public view returns (string memory){    infinite gas
        if(!prof){
            return "You cannot enter College";
        }else {
            return "You can enter College";        }
    }
```

OUTPUT:



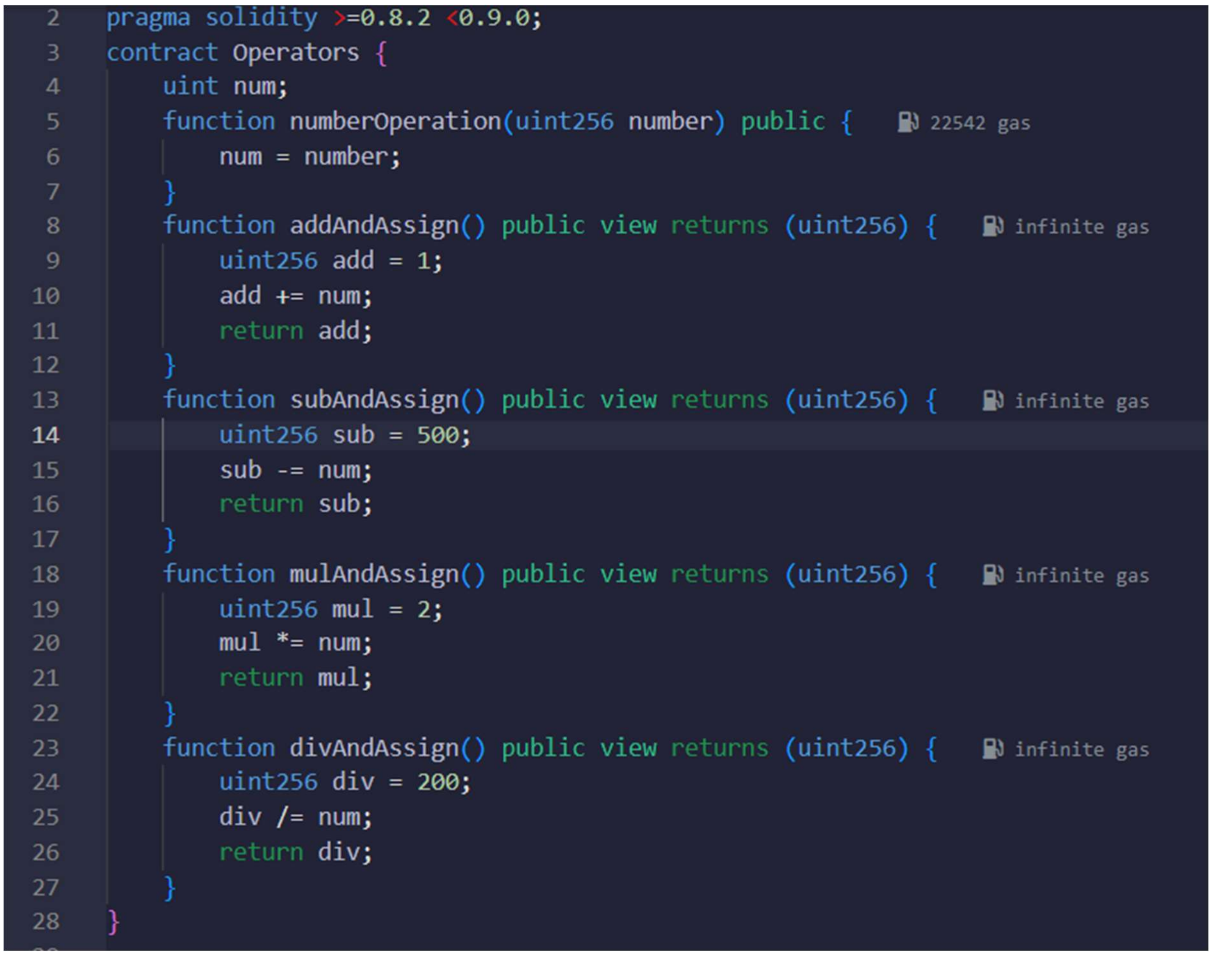
```
Deployed/Unpinned Contracts
OPERATORS AT 0XD7A...F771B (I
Balance: 0 ETH
getDetails  true,true
andOp
0: string: You can take admission
notOp
0: string: You can enter College
orOp
0: string: You can take part in extra curricular activities.
```



```
Deployed/Unpinned Contracts
OPERATORS AT 0XD7A...F771B (I
Balance: 0 ETH
getDetails  true,false
andOp
0: string: You cannot take admission.
notOp
0: string: You cannot enter College
orOp
0: string: You can take part in extra curricular activities.
```

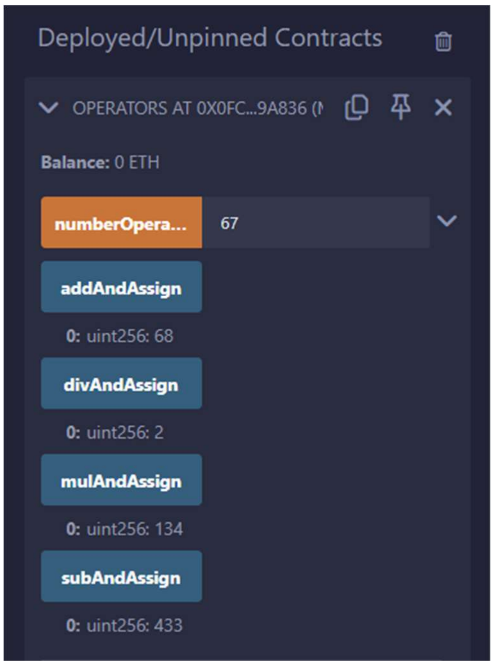
[C] Assignment operator (+=, -=, *=, /=)

CODE:



```
pragma solidity >=0.8.2 <0.9.0;
contract Operators {
    uint num;
    function numberOperation(uint256 number) public {    22542 gas
        num = number;
    }
    function addAndAssign() public view returns (uint256) {    infinite gas
        uint256 add = 1;
        add += num;
        return add;
    }
    function subAndAssign() public view returns (uint256) {    infinite gas
        uint256 sub = 500;
        sub -= num;
        return sub;
    }
    function mulAndAssign() public view returns (uint256) {    infinite gas
        uint256 mul = 2;
        mul *= num;
        return mul;
    }
    function divAndAssign() public view returns (uint256) {    infinite gas
        uint256 div = 200;
        div /= num;
        return div;
    }
}
```

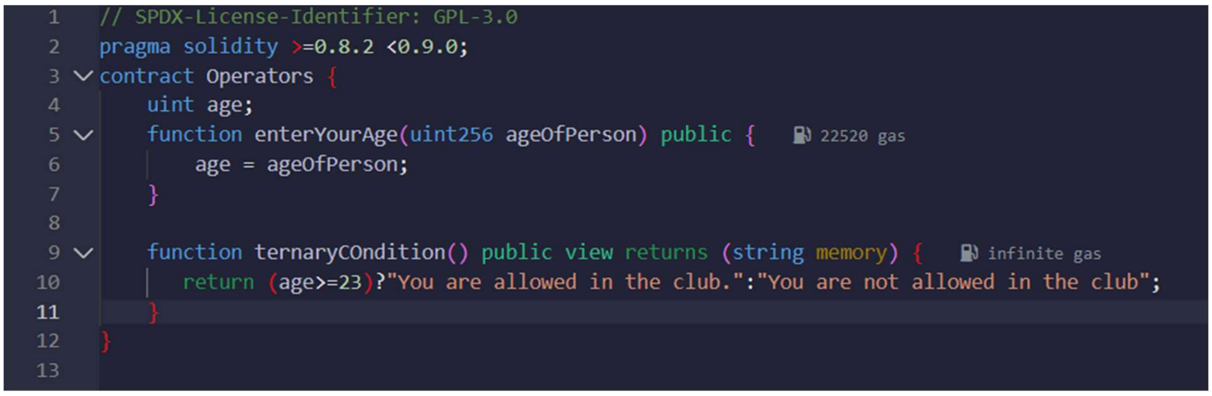
OUTPUT:



```
Deployed/Unpinned Contracts
OPERATORS AT 0X0FC...9A836 (
Balance: 0 ETH
numberOpera...  67
addAndAssign
0: uint256: 68
divAndAssign
0: uint256: 2
mulAndAssign
0: uint256: 134
subAndAssign
0: uint256: 433
```

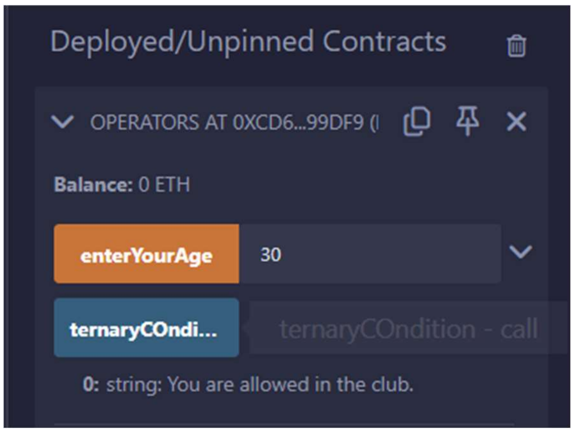
[D] Ternary operator (? :)

CODE:

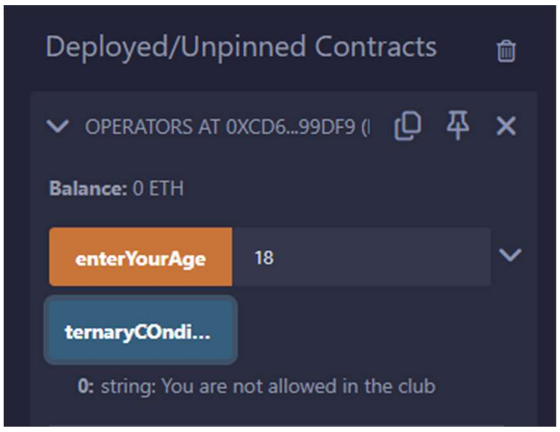


```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract Operators {
    uint age;
    function enterYourAge(uint256 ageOfPerson) public {     22520 gas
        age = ageOfPerson;
    }

    function ternaryCondition() public view returns (string memory) {     infinite gas
       return (age>=23)?"You are allowed in the club.":"You are not allowed in the club";
    }
}
```

OUTPUT:



```
Deployed/Unpinned Contracts
OPERATORS AT 0XCD6...99DF9 (
Balance: 0 ETH
enterYourAge   30
ternaryCOndi...   ternaryCOndition - call
0: string: You are allowed in the club.
```

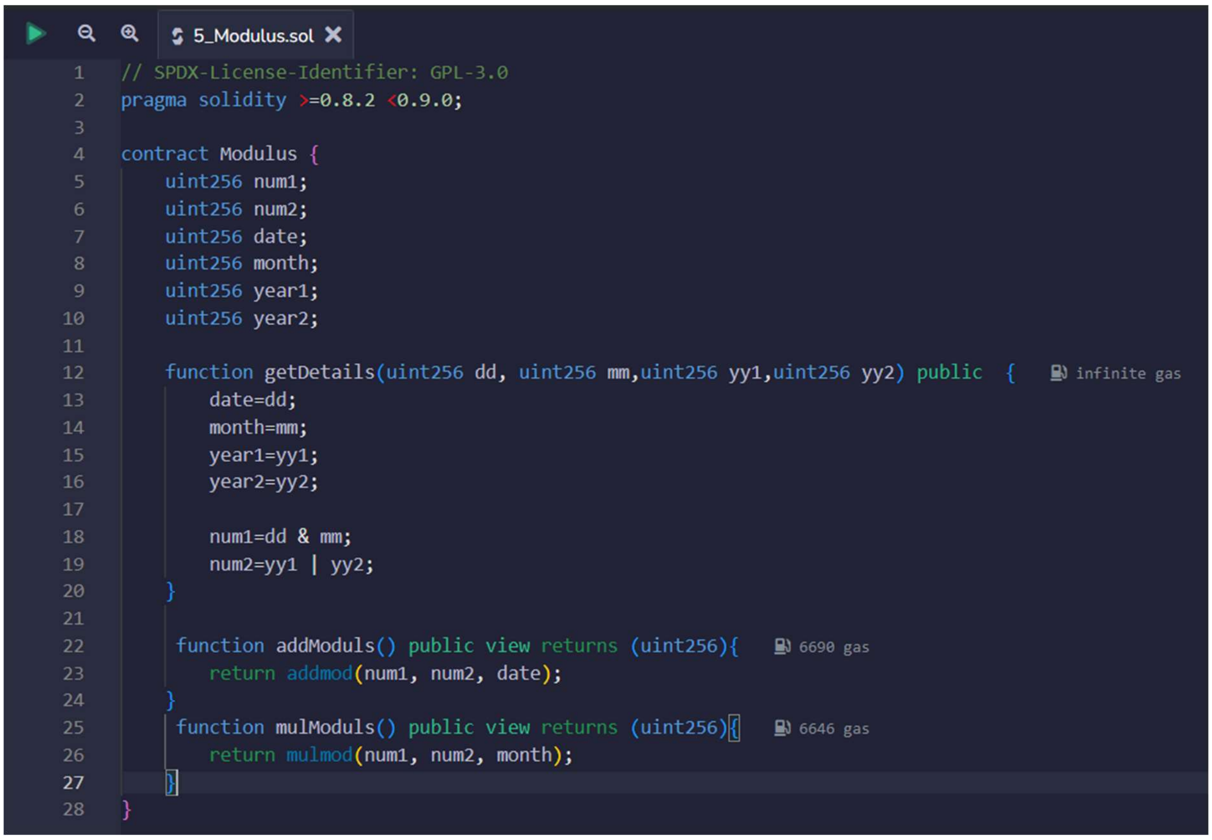


```
Deployed/Unpinned Contracts
OPERATORS AT 0XCD6...99DF9 (
Balance: 0 ETH
enterYourAge   18
ternaryCOndi...
0: string: You are not allowed in the club
```

Practical No 6

AIM : Mathematical functions (mulmod and addmod) and Function overloading.

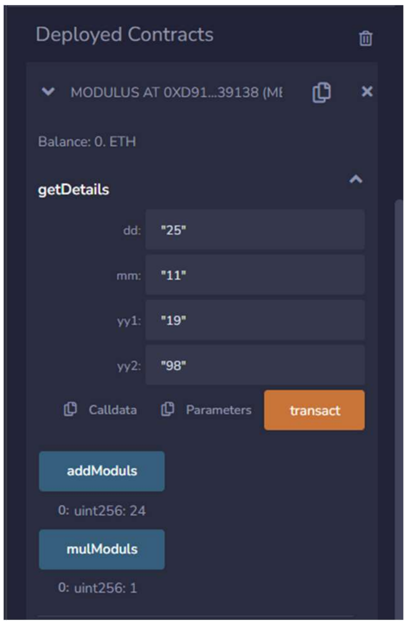
CODE :



5_Modulus.sol

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;

contract Modulus {
    uint256 num1;
    uint256 num2;
    uint256 date;
    uint256 month;
    uint256 year1;
    uint256 year2;

    function getDetails(uint256 dd, uint256 mm,uint256 yy1,uint256 yy2) public  {    infinite gas
        date=dd;
        month=mm;
        year1=yy1;
        year2=yy2;

        num1=dd & mm;
        num2=yy1 | yy2;
    }

     function addModuls() public view returns (uint256){    6690 gas
        return addmod(num1, num2, date);
    }
     function mulModuls() public view returns (uint256){    6646 gas
        return mulmod(num1, num2, month);
    }
}
```

OUTPUT :

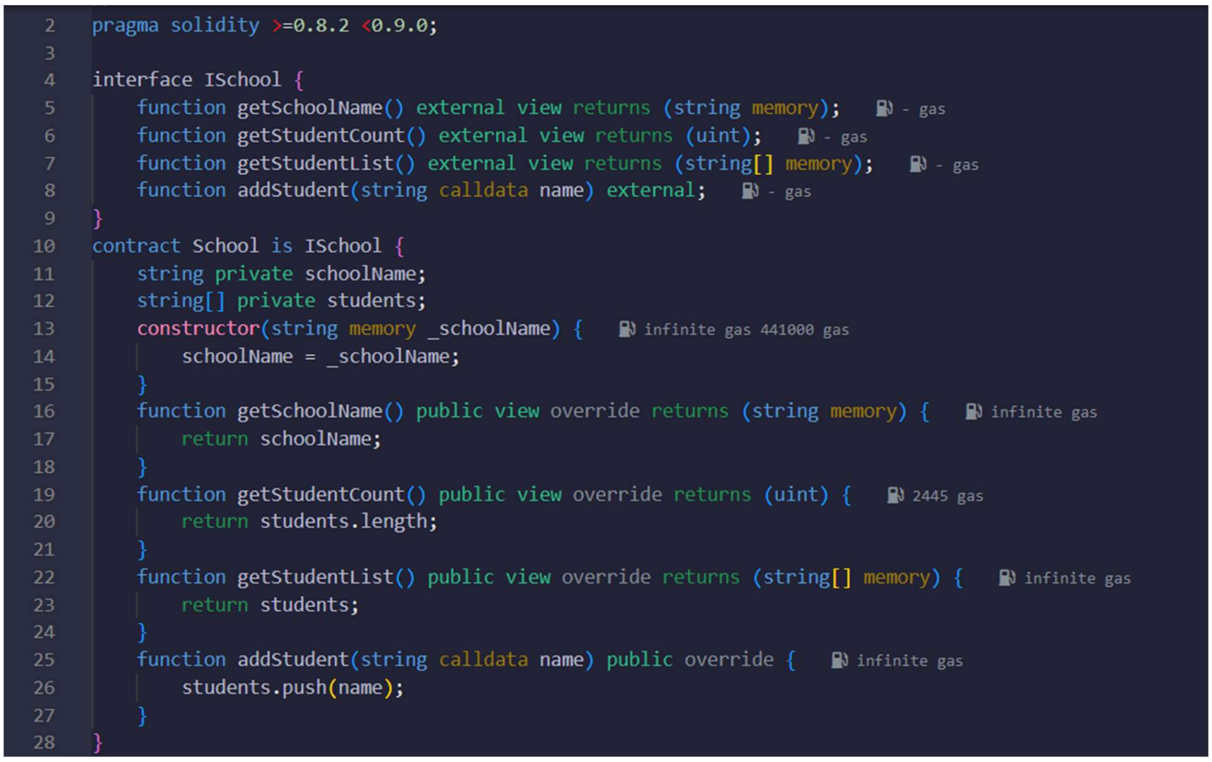


```
Deployed Contracts
MODULUS AT 0XD91...39138 (ME
Balance: 0. ETH
getDetails
dd: "25"
mm: "11"
yy1: "19"
yy2: "98"
Calldata   Parameters   transact
addModuls
0: uint256: 24
mulModuls
0: uint256: 1
```

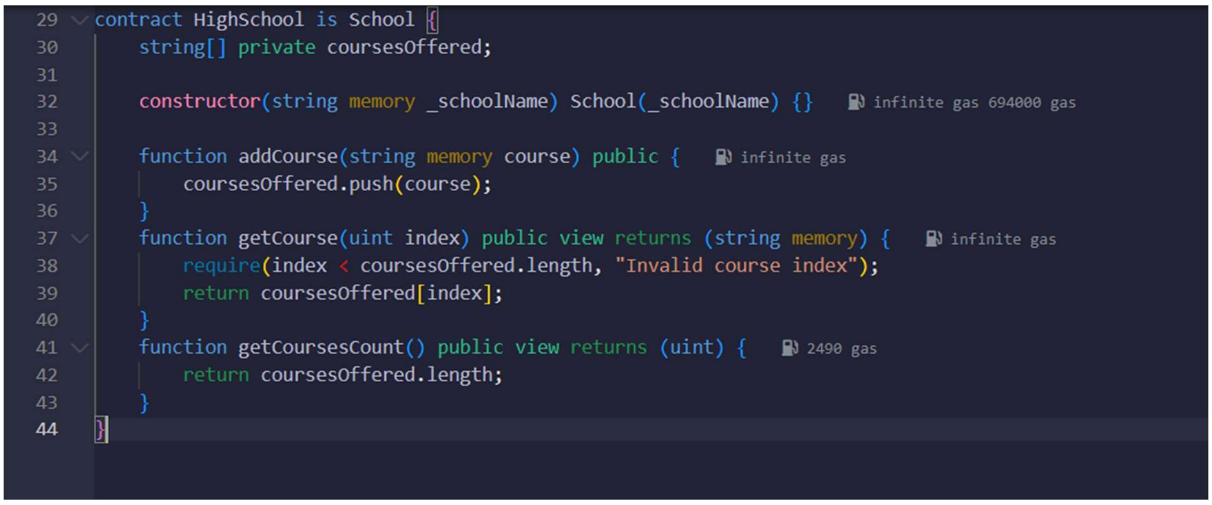
Practical No 7

AIM : Implementation of interface and inheritance.

CODE :

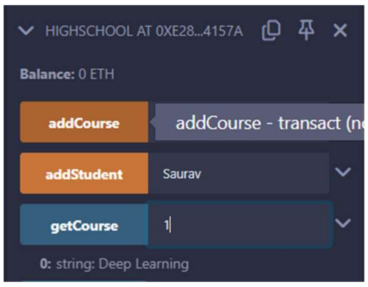


```
pragma solidity >=0.8.2 <0.9.0;

interface ISchool {
    function getSchoolName() external view returns (string memory);    - gas
    function getStudentCount() external view returns (uint);    - gas
    function getStudentList() external view returns (string[] memory);    - gas
    function addStudent(string calldata name) external;    - gas
}
contract School is ISchool {
    string private schoolName;
    string[] private students;
    constructor(string memory _schoolName) {    infinite gas 441000 gas
        schoolName = _schoolName;
    }
    function getSchoolName() public view override returns (string memory) {    infinite gas
        return schoolName;
    }
    function getStudentCount() public view override returns (uint) {    2445 gas
        return students.length;
    }
    function getStudentList() public view override returns (string[] memory) {    infinite gas
        return students;
    }
    function addStudent(string calldata name) public override {    infinite gas
        students.push(name);
    }
}
```

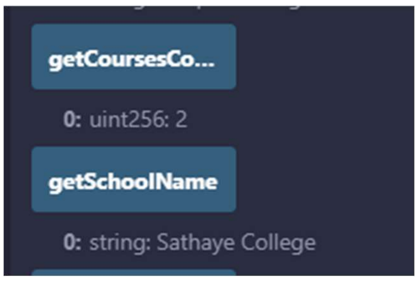


```
contract HighSchool is School {
    string[] private coursesOffered;

    constructor(string memory _schoolName) School(_schoolName) {}    infinite gas 694000 gas

    function addCourse(string memory course) public {    infinite gas
        coursesOffered.push(course);
    }
    function getCourse(uint index) public view returns (string memory) {    infinite gas
        require(index < coursesOffered.length, "Invalid course index");
        return coursesOffered[index];
    }
    function getCoursesCount() public view returns (uint) {    2490 gas
        return coursesOffered.length;
    }
}
```

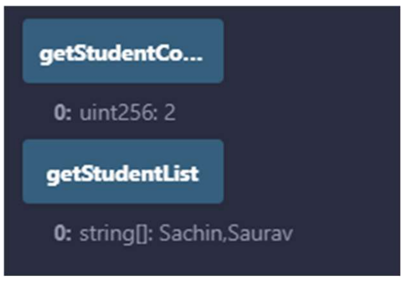
OUTPUT :



```
HIGHSCHOOL AT 0XE28...4157A
Balance: 0 ETH
addCourse    addCourse - transact (n
addStudent   Saurav
getCourse    1
0: string: Deep Learning
```



```
getCoursesCo...
0: uint256: 2
getSchoolName
0: string: Sathaye College
```



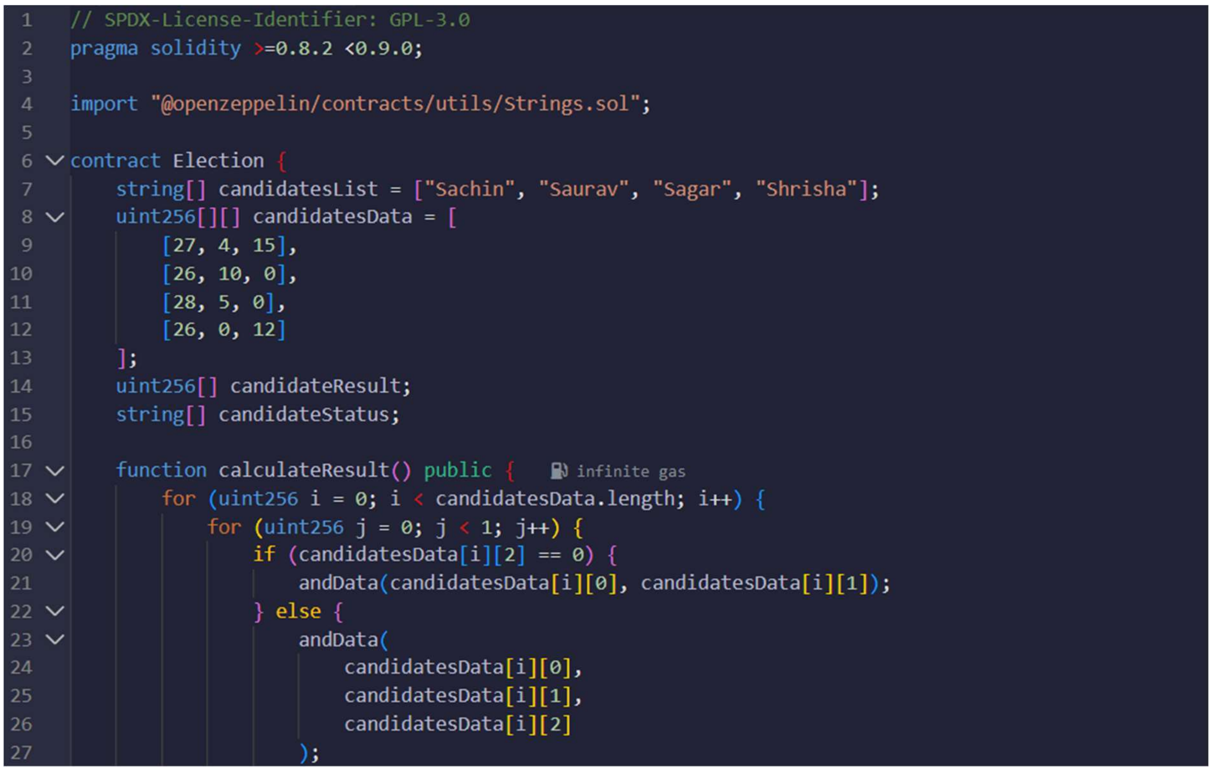
```
getStudentCo...
0: uint256: 2
getStudentList
0: string[]: Sachin,Saurav
```

SACHIN BHOJA ACHARYA
Roll No: 01

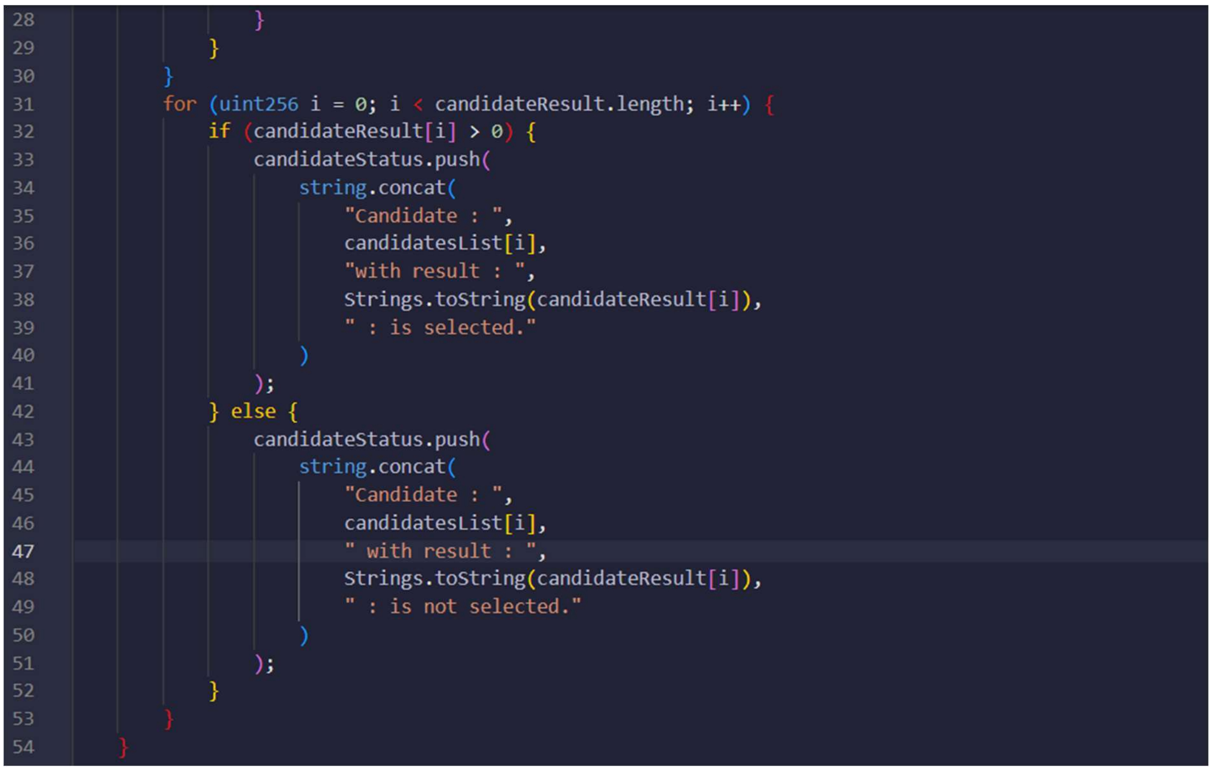
Practical No 8

AIM : Selection of candidate in election.

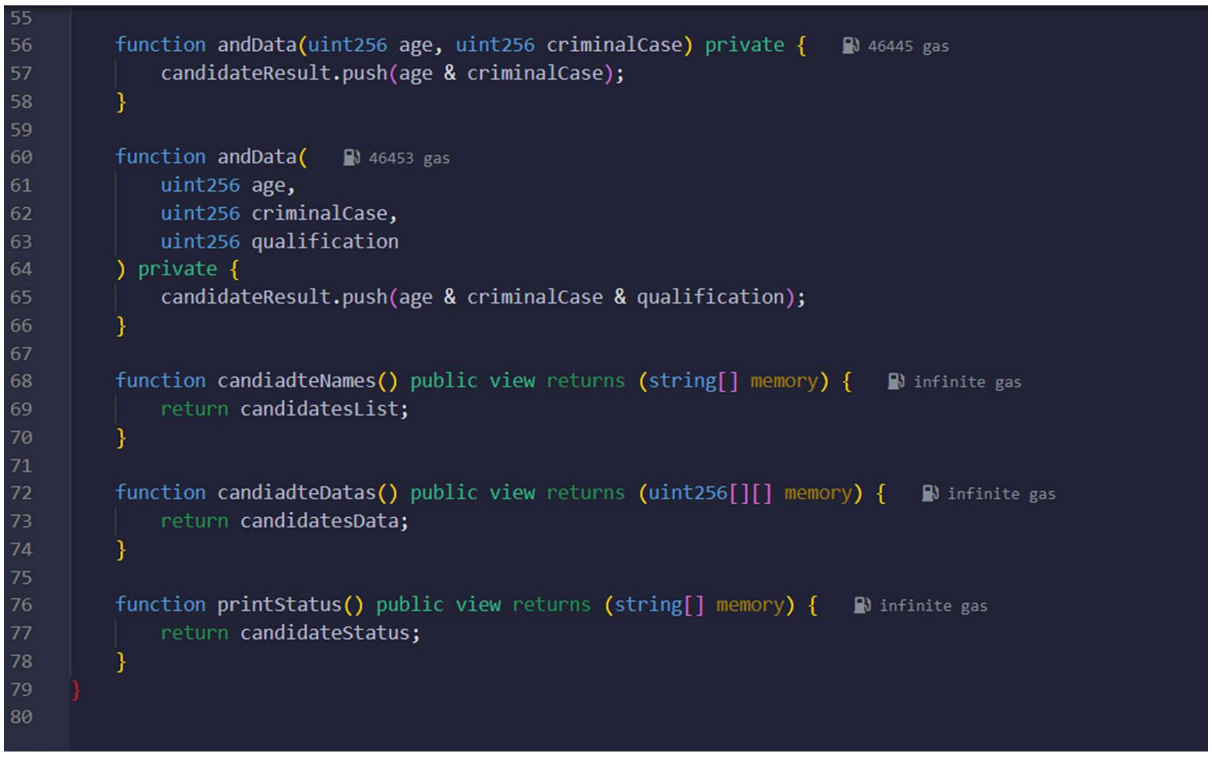
CODE :



```solidity
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;

import "@openzeppelin/contracts/utils/Strings.sol";

contract Election {
    string[] candidatesList = ["Sachin", "Saurav", "Sagar", "Shrisha"];
    uint256[][] candidatesData = [
        [27, 4, 15],
        [26, 10, 0],
        [28, 5, 0],
        [26, 0, 12]
    ];
    uint256[] candidateResult;
    string[] candidateStatus;

    function calculateResult() public {
        for (uint256 i = 0; i < candidatesData.length; i++) {
            for (uint256 j = 0; j < 1; j++) {
                if (candidatesData[i][2] == 0) {
                    andData(candidatesData[i][0], candidatesData[i][1]);
                } else {
                    andData(
                        candidatesData[i][0],
                        candidatesData[i][1],
                        candidatesData[i][2]
                    );
```

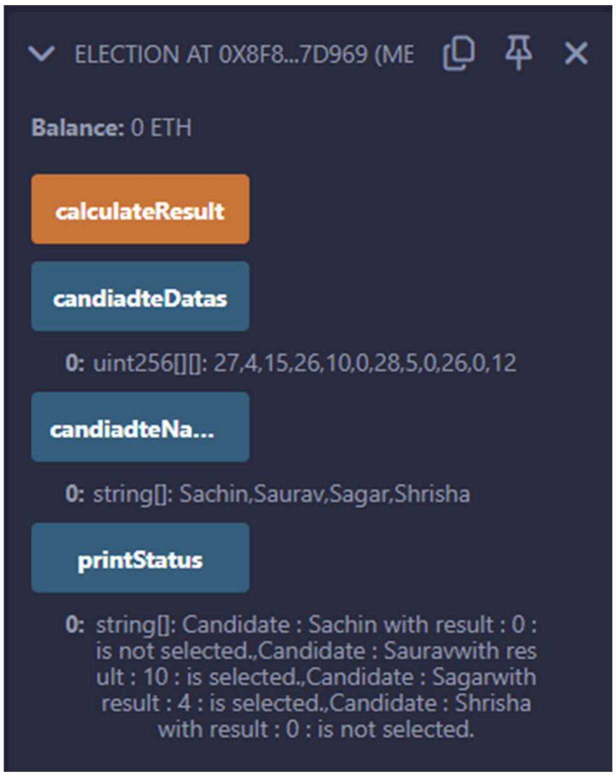


```solidity
                }
            }
        }
        for (uint256 i = 0; i < candidateResult.length; i++) {
            if (candidateResult[i] > 0) {
                candidateStatus.push(
                    string.concat(
                        "Candidate : ",
                        candidatesList[i],
                        "with result : ",
                        Strings.toString(candidateResult[i]),
                        " : is selected."
                    )
                );
            } else {
                candidateStatus.push(
                    string.concat(
                        "Candidate : ",
                        candidatesList[i],
                        " with result : ",
                        Strings.toString(candidateResult[i]),
                        " : is not selected."
                    )
                );
            }
        }
    }
```



```
    function andData(uint256 age, uint256 criminalCase) private {    46445 gas
        candidateResult.push(age & criminalCase);
    }

    function andData(    46453 gas
        uint256 age,
        uint256 criminalCase,
        uint256 qualification
    ) private {
        candidateResult.push(age & criminalCase & qualification);
    }

    function candiadteNames() public view returns (string[] memory) {    infinite gas
        return candidatesList;
    }

    function candiadteDatas() public view returns (uint256[][] memory) {    infinite gas
        return candidatesData;
    }

    function printStatus() public view returns (string[] memory) {    infinite gas
        return candidateStatus;
    }
}
```

<u>OUTPUT</u> :

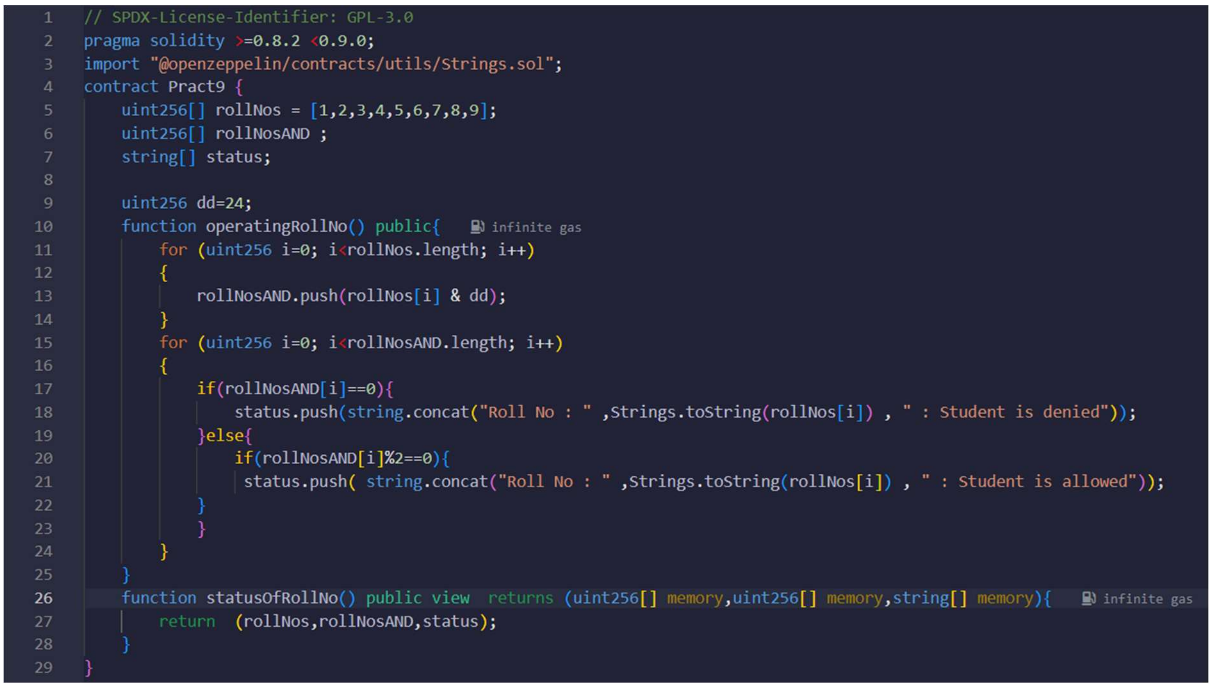


```
Balance: 0 ETH

0: uint256[][]: 27,4,15,26,10,0,28,5,0,26,0,12

0: string[]: Sachin,Saurav,Sagar,Shrisha

0: string[]: Candidate : Sachin with result : 0 :
   is not selected.,Candidate : Sauravwith res
   ult : 10 : is selected.,Candidate : Sagarwith
   result : 4 : is selected.,Candidate : Shrisha
   with result : 0 : is not selected.
```

Practical No 9

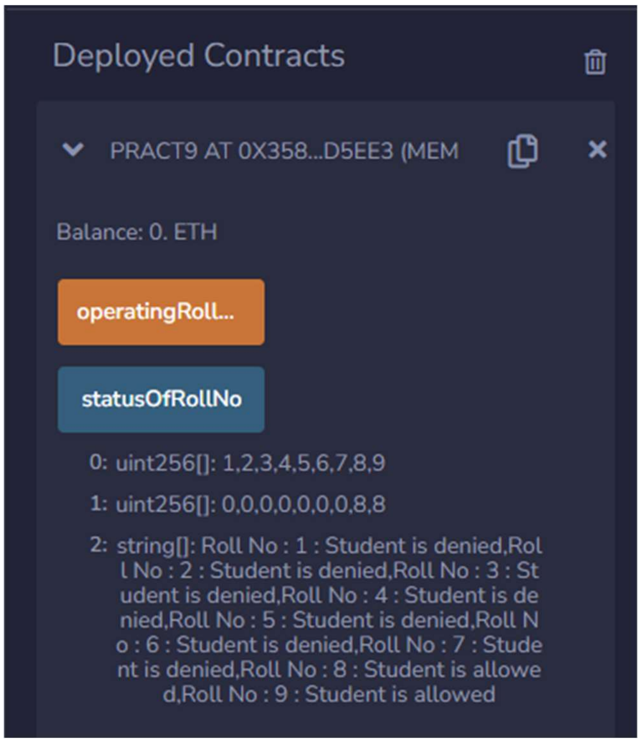
AIM : Write a solidity program to create an array of roll no's and then create a smart contract where it checks the values of the roll no and perform AND operation with today's date DD and if the result is even, then display the message "Student is allowed" else "Denied".

CODE :



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
import "@openzeppelin/contracts/utils/Strings.sol";
contract Pract9 {
    uint256[] rollNos = [1,2,3,4,5,6,7,8,9];
    uint256[] rollNosAND ;
    string[] status;

    uint256 dd=24;
    function operatingRollNo() public{    infinite gas
        for (uint256 i=0; i<rollNos.length; i++)
        {
            rollNosAND.push(rollNos[i] & dd);
        }
        for (uint256 i=0; i<rollNosAND.length; i++)
        {
            if(rollNosAND[i]==0){
                status.push(string.concat("Roll No : " ,Strings.toString(rollNos[i]) , " : Student is denied"));
            }else{
                if(rollNosAND[i]%2==0){
                 status.push( string.concat("Roll No : " ,Strings.toString(rollNos[i]) , " : Student is allowed"));
            }
            }
        }
    }
    function statusOfRollNo() public view  returns (uint256[] memory,uint256[] memory,string[] memory){    infinite gas
        return  (rollNos,rollNosAND,status);
    }
}
```

OUTPUT :

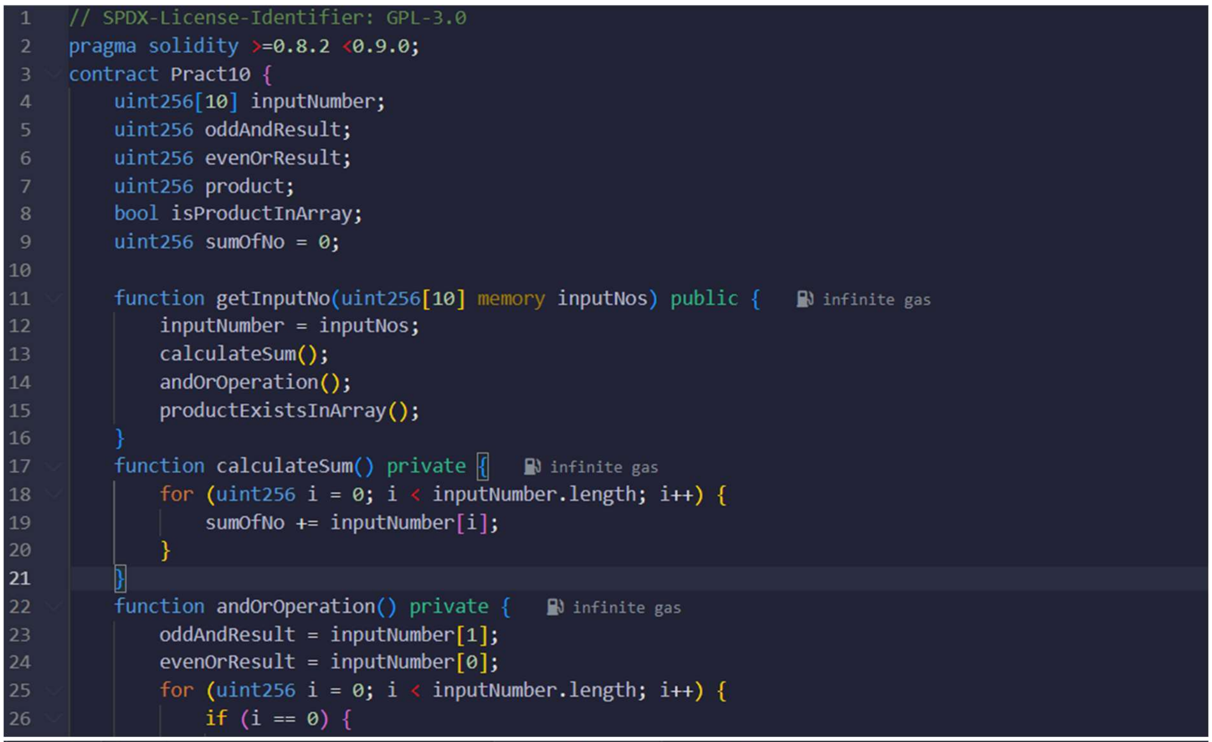


```
0: uint256[]: 1,2,3,4,5,6,7,8,9
1: uint256[]: 0,0,0,0,0,0,0,8,8
2: string[]: Roll No : 1 : Student is denied,Rol
   l No : 2 : Student is denied,Roll No : 3 : St
   udent is denied,Roll No : 4 : Student is de
   nied,Roll No : 5 : Student is denied,Roll N
   o : 6 : Student is denied,Roll No : 7 : Stude
   nt is denied,Roll No : 8 : Student is allowe
   d,Roll No : 9 : Student is allowed
```

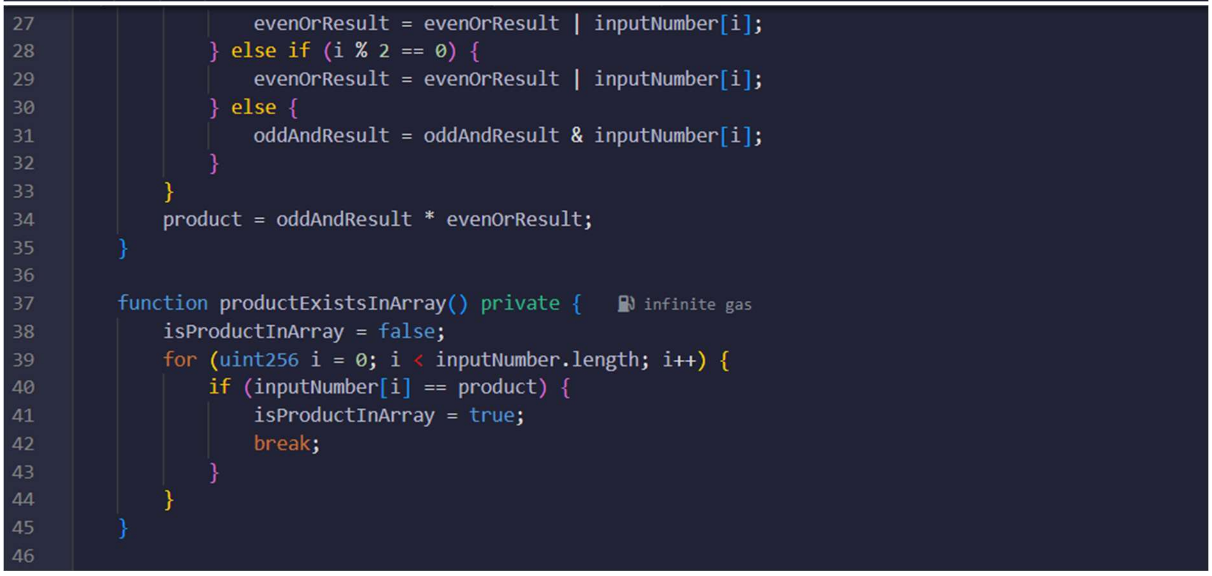
Practical No 10

AIM : Write a solidity program to find the sum of an array of 10 numbers which are taken from the user and then create a smart contract to find the AND operation of Odd positioned numbers and OR operation of Even positioned numbers including 0th Index, hence find the product of the result and also identify whether the result is part of the array or not.

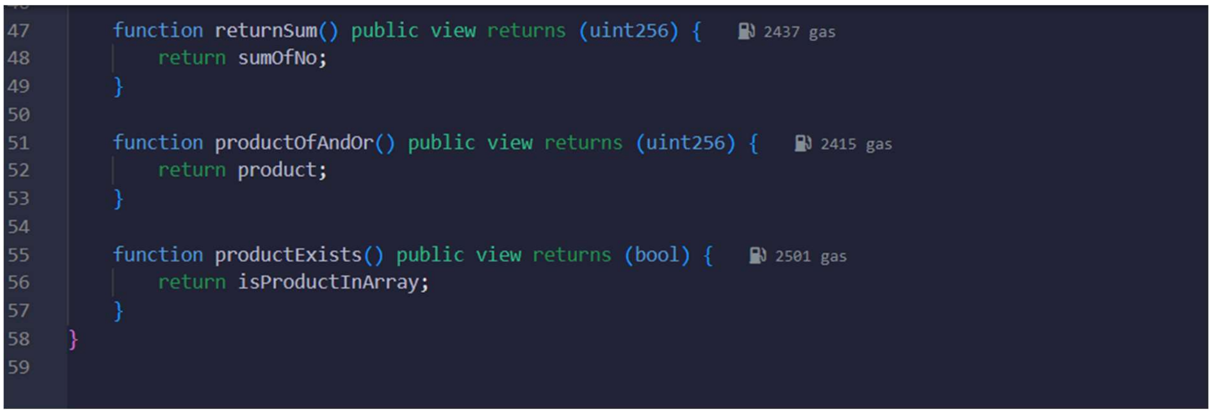
CODE :



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract Pract10 {
    uint256[10] inputNumber;
    uint256 oddAndResult;
    uint256 evenOrResult;
    uint256 product;
    bool isProductInArray;
    uint256 sumOfNo = 0;

    function getInputNo(uint256[10] memory inputNos) public {
        inputNumber = inputNos;
        calculateSum();
        andOrOperation();
        productExistsInArray();
    }
    function calculateSum() private {
        for (uint256 i = 0; i < inputNumber.length; i++) {
            sumOfNo += inputNumber[i];
        }
    }
    function andOrOperation() private {
        oddAndResult = inputNumber[1];
        evenOrResult = inputNumber[0];
        for (uint256 i = 0; i < inputNumber.length; i++) {
            if (i == 0) {
```

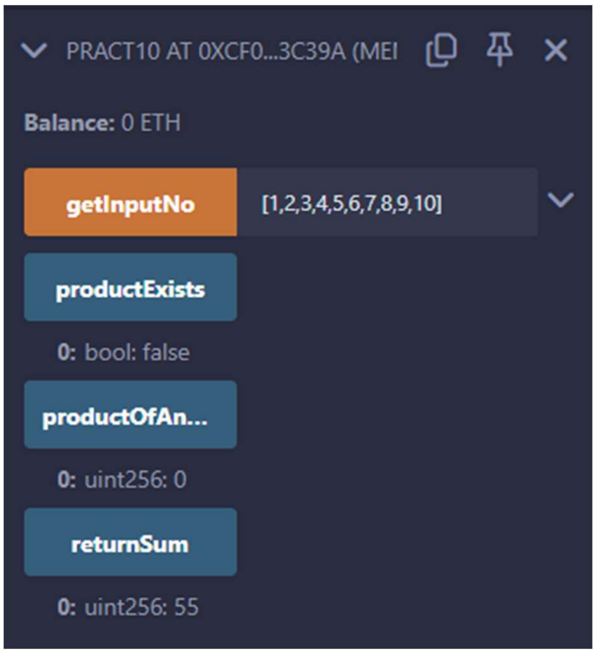


```
                evenOrResult = evenOrResult | inputNumber[i];
            } else if (i % 2 == 0) {
                evenOrResult = evenOrResult | inputNumber[i];
            } else {
                oddAndResult = oddAndResult & inputNumber[i];
            }
        }
        product = oddAndResult * evenOrResult;
    }

    function productExistsInArray() private {
        isProductInArray = false;
        for (uint256 i = 0; i < inputNumber.length; i++) {
            if (inputNumber[i] == product) {
                isProductInArray = true;
                break;
            }
        }
    }

```



```
    function returnSum() public view returns (uint256) {    2437 gas
        return sumOfNo;
    }

    function productOfAndOr() public view returns (uint256) {    2415 gas
        return product;
    }

    function productExists() public view returns (bool) {    2501 gas
        return isProductInArray;
    }
}

```

OUTPUT :

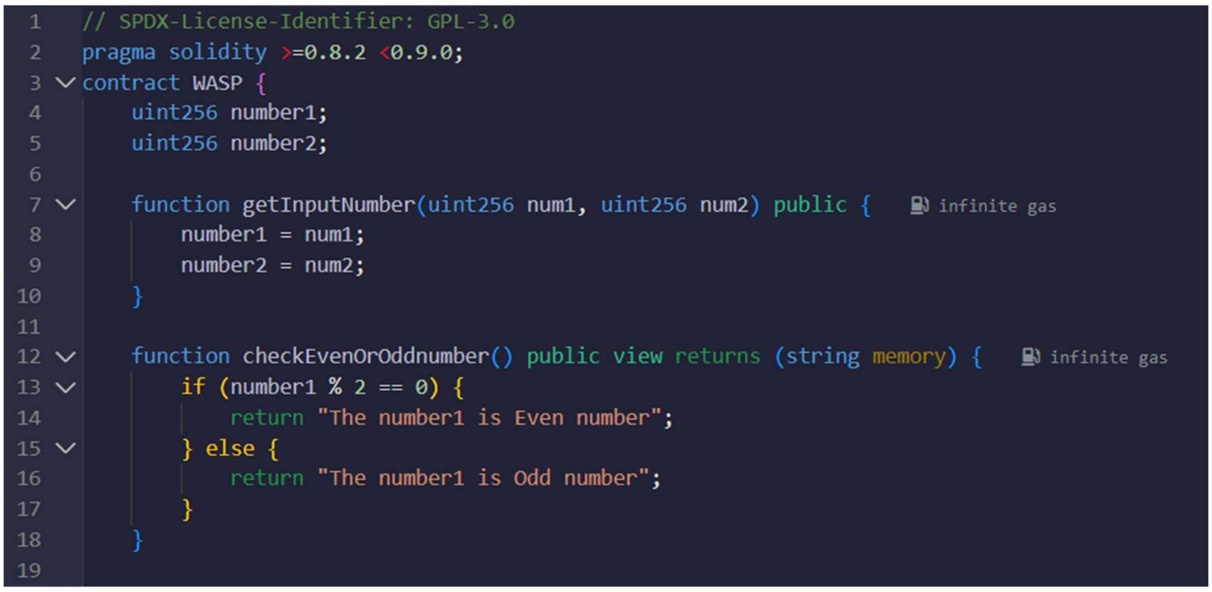


```
PRACT10 AT 0XCF0...3C39A (MEI

Balance: 0 ETH

getInputNo    [1,2,3,4,5,6,7,8,9,10]

productExists
0: bool: false

productOfAn...
0: uint256: 0

returnSum
0: uint256: 55
```

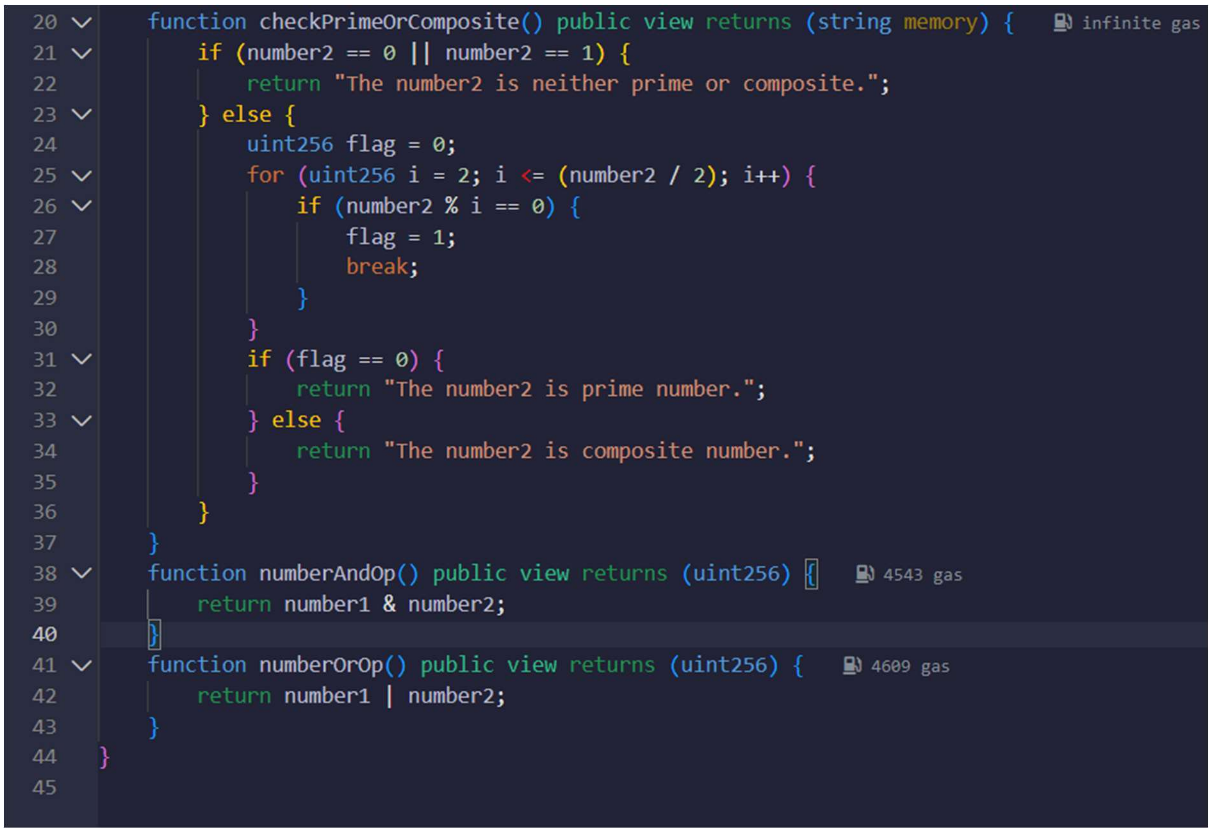
Practical No 11

AIM : Write a solidity program to find whether a number is even or odd and another number is prime or composite. Also find the AND and OR operation of the two numbers.

CODE :

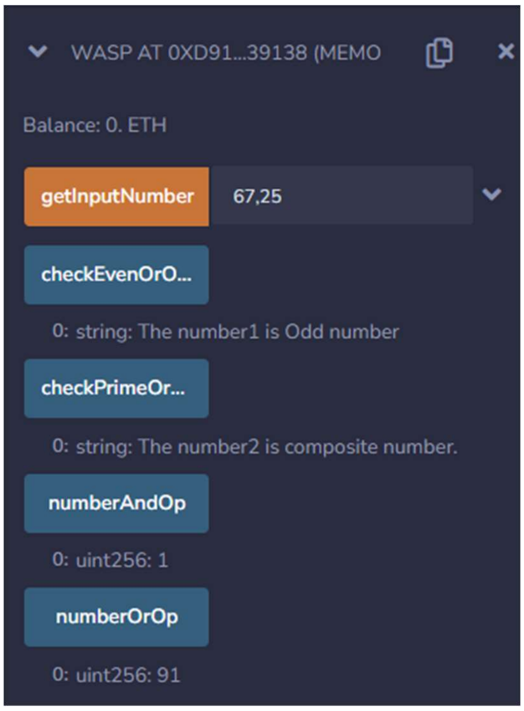


```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract WASP {
    uint256 number1;
    uint256 number2;

    function getInputNumber(uint256 num1, uint256 num2) public {    infinite gas
        number1 = num1;
        number2 = num2;
    }

    function checkEvenOrOddnumber() public view returns (string memory) {    infinite gas
        if (number1 % 2 == 0) {
            return "The number1 is Even number";
        } else {
            return "The number1 is Odd number";
        }
    }

```



```
    function checkPrimeOrComposite() public view returns (string memory) {    infinite gas
        if (number2 == 0 || number2 == 1) {
            return "The number2 is neither prime or composite.";
        } else {
            uint256 flag = 0;
            for (uint256 i = 2; i <= (number2 / 2); i++) {
                if (number2 % i == 0) {
                    flag = 1;
                    break;
                }
            }
            if (flag == 0) {
                return "The number2 is prime number.";
            } else {
                return "The number2 is composite number.";
            }
        }
    }
    function numberAndOp() public view returns (uint256) {    4543 gas
        return number1 & number2;
    }
    function numberOrOp() public view returns (uint256) {    4609 gas
        return number1 | number2;
    }
}

```

OUTPUT :



```
Balance: 0. ETH
getInputNumber  67,25
0: string: The number1 is Odd number
0: string: The number2 is composite number.
0: uint256: 1
0: uint256: 91
```